Hrubec, Thomas

Rechnerunterstützte Angebotserstellung veranschaulicht anhand von Beispielen im Klimaanlagenbau

DIPLOMARBEIT

HOCHSCHULE MITTWEIDA (FH)

UNIVERSITY OF APPLIED SCIENCES

Technische Informatik

Mittweida, 2009

Hrubec, Thomas

Rechnerunterstützte Angebotserstellung veranschaulicht anhand von Beispielen im Klimaanlagenbau

eingereicht als

DIPLOMARBEIT

an der

HOCHSCHULE MITTWEIDA (FH)

UNIVERSITY OF APPLIED SCIENCES

Technische Informatik

Erstprüfer : Prof. Ing. Dr. Jürgen Ruck

Zweitprüfer : Dipl. Ing. (FH) Peter Hrubec

Bibliographische Beschreibung

Hrubec, Thomas:
Rechnerunterstützte Angebotserstellung veranschaulicht anhand von Beispielen im Klimaanlagenbau - 2009 - 190 S.
Mittweida, Hochschule Mittweida (FH), Fachbereich Informationstechnik & Elektrotechnik
Diplomarbeit, 2009

Referat:

Ziel der Diplomarbeit ist es, eine Arbeitsgrundlage in Form eines Programms zu schaffen, mit dessen Hilfe eine lückenlose Angebotserstellung von Projekten ermöglicht wird. Im Speziellen sollen Angebote im Bereich des Klimaanlagenbaus, inklusive der kompletten Mess-, Steuer- und Regelungstechnik, erstellt werden. Um die Verständlichkeit zu erhöhen, wird ein reales Beispiel einer Klimaanlagenregelung eines Kraftwerkes herangezogen. Zum Abschluss erfolgen weitere Beispiele aus der Praxis und ein Ausblick auf die zukünftige Nutzung dieses Programms.

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abb.	Abbildung
AP	Anwendungsprogramm
BDE	Borland Database Engine
Bsp.	Beispiel
bzgl.	bezüglich
bzw.	beziehungsweise
CD	Compact Disk
COM	Component Object Model
CPU	Control Processing Unit
d.h.	das heißt
DDC	Direct Digital Control
DBMS	Datenbankmanagementsystem
etc.	et cetera (und so weiter)
f	folgende
ff	folgenden
F&G	Fühler und Geber
GLT	Gebäudeleittechnik
HVAC	Heating Ventilation Air Conditioning
IBS	Inbetriebsetzung
lt.	laut
LV	Leistungsverzeichnis
S.	Seiten
SPS	speicherprogrammierbare Steuerung
SS	Schaltschrank
SSBG	Beistellgeräte Schaltschrank
SQL	Structured Query Language
SW	Software
Tab.	Tabelle
u.a.	unter anderem

usw.	und so weiter
VCL	Visual Component Library
Vgl.	vergleiche
z.B.	zum Beispiel

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

Diese Diplomarbeit hat das Ziel, die Angebotslegung elektronisch zu unterstützen. Es soll ein einfaches und schnelles Arbeiten gewährleistet sein, um möglichst rasch ein Angebot, im Speziellen für mess-, steuer- und regeltechnische Anlagen des Klimabaus erstellen zu können. Nach Sichtung des zu erwartenden Programmieraufwandes, beschloss der Autor, ein für ihn unbekanntes Programmiertool zu verwenden. Erwartungsgemäß ergab sich daraus eine große Herausforderung, aber der Verfasser entschied sich diese Programmierung anzueignen. Für sein weiteres Berufsleben bei der Firma KBA - Mödling AG ist ihm die Kenntnis über diese Programmiersprache von Nutzen. Die Auswahl fiel auf das Programmiertool „C++ Builder 5.0“ von Borland.
Diese Arbeit befasst sich damit, wie aufwendig es ist, die Programmierung zu erlernen, welche Probleme bei der Entwicklung dieser Arbeit auftraten und wie nützlich dieses Programm schlussendlich ist.

Die Anwendung soll in einer Firma eingesetzt werden, welche Klimaanlagen beispielsweise in Kraftwerken realisiert. So ein Projekt besteht aus mehreren Teilen. Diese Teile werden wie folgt aufgegliedert:

- Anfrage
- Angebotserstellung inkl. Kalkulation
- Zuschlag
- Projektdurchführung
- Inbetriebnahme
- Dokumentation

Ein wesentlicher Teil des gesamten Projekts ist die Angebotserstellung inklusive Kalkulation. Genau diesem Teil widmet sich diese Arbeit, um das Angebot rechnerunterstützt entwickeln zu können.

1.2 Kapitelübersicht

Kapitel 1 umfasst die Einleitung und Motivation dieser Arbeit.

Kapitel 2 beschreibt die Situationsanalyse zur Angebotserstellung sowie deren Grundlagen.

Kapitel 3 befasst sich mit dem Entwurf und der Zieldefinition dieser Arbeit.

Kapitel 4 beinhaltet die Implementierung, sowie die Umsetzung und Entwicklung der Applikation basierend auf der Zieldefinition.

Kapitel 5 beschäftigt sich mit der Anwendung und beschreibt deren Verwendung. Zusätzlich wird ein reales Projekt mit dieser Anwendung erstellt und kalkuliert.

Kapitel 6 enthält eine Zusammenfassung der Ergebnisse, sowie Ausblicke auf Verwendung, Umsetzung und Weiterentwicklung dieser Applikation.

Anhang A beinhaltet den Quelltext der entwickelten Anwendung.

Anhang B umfasst den Quelltext der Klasse TKalkulation.

2 Allgemeines zur Angebotserstellung

Um ein Angebot erstellen zu können, müssen zuerst viele Dinge geklärt werden. Man darf nicht vergessen, dass Unternehmen oft sehr viele Angebote erstellen, aber davon nur einen Bruchteil an Aufträgen erhalten. Man spricht von einem branchenbezogenem Verhältnis zwischen 3 und 10 Prozent den Auftrag zu erhalten[1]. Jedes Unternehmen darf die entstehenden Kosten, welche durch eine Angebotserstellung auftreten, aber zu keinem Auftrag führen, nicht vernachlässigen. Ziel jedes Unternehmens ist diese unproduktiven Kosten so gering wie möglich zu halten.

2.1 Wann muss ein Angebot erstellt werden

Im technischen Vertrieb, wo es komplexe Anlagen gibt, werden in der Regel individuelle Angebote erstellt. Bei den Produktklassen B und C (siehe Tabelle 1) hat ein Angebot keine große Bedeutung, da hier im Normalfall auf standardisierte Maschinen oder Produkte zurückgegriffen wird. Die mögliche Ausstattung ist hierbei in Katalogen mit Fixpreisen festgelegt.
Geht es aber um rein kundenspezifische, komplexe Lösungen, wie z.B. im Anlagenbau (Produktklasse A), so werden diese Projekte über Angebote genauer definiert. Man kann nicht von fertigen Komplettlösungen ausgehen, da jede Anlage seine Eigenheiten und Schwerpunkte hat und daher kein Projekt dem anderen gleicht.

[1] Beichter, Siegfried: Rechnergestützte Technische Problemlösung bei der Angebotserstellung von Flexiblen Drehzellen, Universität Karlsruhe

	A	**B**	**C**
Produktklasse	Komplexe Anlagentechnik mit individueller Anpassung	Konkretisierbare Maschinen ohne individueller Anpassung	Reine Beschaffungsgüter für laufende Fertigung
Beispiel	Klimaanlage im Anlagenbau	Druckmaschine	Roh-/Hilfs-, Betriebsstoffe
Entscheidungsträger	Techn. Leitung Produktion Einkauf	Techn. Leitung Kaufm. Leitung Einkauf	 Einkauf
Dauer Entscheidung	>3 Monate	1- 3 Monate	< 1 Monat
Technisches Fachwissen des Verkäufers	hoch	mittel	gering
Erklärungsbedarf, technische Beratung	hoch	mittel	gering
Aufwand zur Angebotserstellung	hoch	mittel	gering
- Technische Vorklärung	muss	kann	----------
- Projektierung	muss	kann	----------
- Konstruktion	individuell	vorgegebene Ausstattungsvarianten	Katalog
- Kalkulation Preisfindung	individuell	Preisliste	Preisliste
- Präsentation	muss	kann	----------

Tab. 1: Anforderungen an die Angebotserstellung[2]

[2] Schwetz, Dipl.-Betriebsw. W: Computerunterstützte Angebotserstellung im Rahmen integrierter Vertriebssteuerungssysteme, 1990, Pfinztal, S136

2.2 Zielsetzung des Angebots

Die Zielsetzung eines jeden Angebotes sollte sein, den Auftrag zu erhalten. Um dies zu erreichen, sollte nur bei realistischen Chancen auf den Zuschlag, Zeit und Arbeit in das Angebot investiert werden. Wird in einem Unternehmen ein Angebot erstellt, so durchläuft es oft mehrere Abteilungen, wie Konstruktionsabteilung, Disposition, Einkauf usw. bis erst das komplette Angebot dem Kunden vorgelegt werden kann. Dies ist natürlich sehr zeitaufwendig und kostspielig. Ziel sollte sein, diese Faktoren sehr gering zu halten.
Weitere Ziele sollten eine größere Anzahl an abgegebenen Angeboten, schnellere Reaktion bei Anfragen und Angebotsversand und ein einheitliches, attraktives äußeres Erscheinungsbild sein, um dem Kunden durch rasches professionelles Auftreten zu überzeugen.[3]
Durch den zunehmenden internationalen Wettbewerb ist der Qualitätsanspruch für Angebote sehr gestiegen. Trotz geringen, aufwandsarmen Arbeitseinsatzes soll ein vollständiges, qualitatives Angebot erstellt werden.

2.3 Die Lösungen aller Erwartungen

Durch eine rechnerunterstützte Angebotserstellung sollen folgende Punkte umgesetzt werden können:

- Verringerung des Arbeitsaufwandes (Kostensenkung)
- Verringerung der Fehlerhäufigkeit in Angeboten
- Qualitativ hochwertigere Angebote erstellen
- Minimierung des Erstellungsaufwands durch Rückgriff auf bereits vorhandene Lösungen, welche nur noch modifiziert werden müssen
- Erhöhung der Auftragswahrscheinlichkeit

[3] Kirsten, Dr. J.: Strategische Bedeutung von Bertriebsinformationsunterstützung, Wiesbaden, 1990

2.4 Der Weg

Der Weg zu dieser Lösung wird in folgender Arbeit verwirklicht. Die komplette Angebotsbearbeitung, zu deren Aufgaben auch eine Adressverwaltung und die Kalkulation gehören, soll in einem Programm erfolgen. Die formschöne Ausgabe vollendet das Programm. Zu jedem Zeitpunkt ist es möglich, die Erstellung zu unterbrechen, abzuspeichern und zu einem späteren Zeitpunkt weiter zu bearbeiten. Genaueres siehe auch Kapitel 5.

2.5 Erfolg von Angeboten

Der Erfolg von Angeboten hängt von mehreren Faktoren ab. So spielen die Markttransparenz, die Konkurrenzsituation, die Produktqualität, die Dauer der Anfragebearbeitung und der Liefertermin eine große Rolle. Der Preis aber ist natürlich der bestimmendste Faktor für die Auftragsvergabe.[4]

Markttransparenz: Manche Kunden wünschen eine hohe Transparenz, d.h. je mehr Informationen, wie Preis, Herkunft, Lieferung und sonstige Konditionen über alle verwendeten Geräte und Bauteile vorliegen, umso transparenter ist das Angebot.

Konkurrenzsituation: Je mehr Mitanbieter es auf dem Markt gibt, desto schwieriger wird es, Bestbieter mit guten Preisen zu werden.

Produktqualität: Diese ist ein entscheidender Faktor für die meisten Kunden. Viele wünschen sich Markenprodukte, da sie von deren Qualität überzeugt sind.

Dauer der Anfragebearbeitung: Bei jeder Anfrage gibt es Fristen, innerhalb welcher das Angebot eingereicht sein muss. Ist das Unternehmen aber so sehr ausgelastet, dass es nicht möglich ist, diese Frist einzuhalten, ist dieser Auftrag verloren.

Liefertermin: Wird vom Kunden ein Liefertermin gewünscht, welcher schwierig einzuhalten ist, so kann dies natürlich auch entscheidend sein, ob der Anbieter diesen etwaigen Auftrag annehmen kann.

Preis: Einer der wesentlichsten Faktoren, da jeder Auftraggeber das bestmöglichste Angebot für den kleinstmöglichen Preis erhalten möchte.

---

[4] Feller Achim H.: Kalkulation in der Angebotsphase mit dem selbsttätig abgeleiteten Erfahrungswissen der Arbeitsplanung, 1992, S2 f

In der Regel gibt es 2 Angebotsrisiken:[5]

1. Man erhält keinen Auftrag, aufgrund eines zu hohen Preises
2. Der Auftrag wird erteilt, aber der kalkulierte Preis reicht nicht, um den gewünschten Gewinn oder sogar den nötigen Deckungsbeitrag bei diesem Projekt zu erzielen

Gerade diese Punkte veranschaulichen, dass das Angebot sehr gut überlegt sein muss. Um gute Zuschlagschancen zu erhalten, muss kostendeckend kalkuliert werden und der Angebotspreis muss akzeptabel sein.

2.6 Angebotserstellung

Diese kann natürlich händisch erfolgen, was mehrere Nachteile mit sich bringen kann:

- Höherer Zeitaufwand
- Erstellung attraktiver Angebote nur mit hohem Arbeitsaufwand
- Mühsames Durchforsten von Geräte- und Produktkatalogen usw.

Mit dieser Software können alle diese Nachteile entschärft und eine höhere Produktivität erzielt werden. Es kann schneller auf Anfragen reagiert werden, es können mehrere Angebote in derselben Zeit abgegeben werden und das äußere Erscheinungsbild ist immer identisch. All diese Punkte ermöglichen einen größeren Auftragseingang bei dem Unternehmen.

2.7 Mitarbeiterakzeptanz

Um das neue Angebotserstellungsprogramm auch optimal nützen zu können, müssen die Mitarbeiter überzeugt sein, dass dieses Programm ihre Arbeit erleichtert und sie keine zusätzlichen Aufgaben aufgebürdet bekommen.

---

[5] Feller Achim H.: Kalkulation in der Angebotsphase mit dem selbsttätig abgeleiteten Erfahrungswissen der Arbeitsplanung, 1992, S3

Folgende Punkte sind dabei entscheidend[6]:

- Klarer Nutzen für den Mitarbeiter
- Weniger Verwaltungsaufwand
- Bedienerfreundliches Programm
- Ausgiebige Schulung und Anwendungsbetreuung
- Teilnahme bei der Anforderungsgestaltung
- Klare und vollständige Dokumentation

2.8 Grundlagen zur Angebotserstellung

Grobgliederung:

Wird ein Angebot erstellt, so kann dieses grob in 3 grundlegende Abschnitte eingeteilt werden:

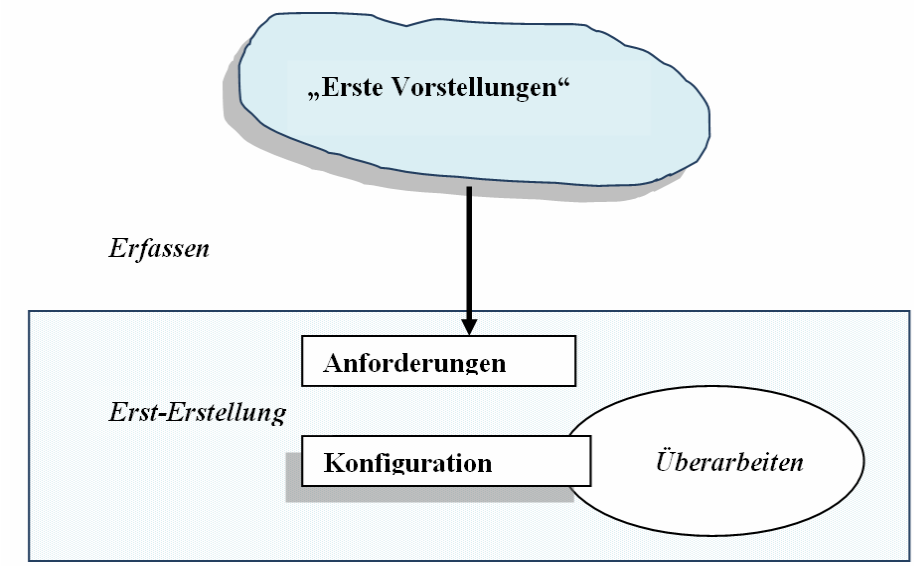


Abb. 1: Werdegang eines Angebots[7]

[6] Kirsten, Dr. J.: Strategische Bedeutung von Bertriebsinformationsunterstützung, Wiesbaden, 1990

[7] Hölzler, Dipl.-Ing. E: Kraus, Dipl.-Ing. H.: Konfiguration von System-Produkten – von der Anforderung zu Angebot und Stückliste, München, 1990, S66

Nach Äußerung des Kunden über seine „ersten Vorstellungen“ werden diese erfasst und die Anforderungen dokumentiert. Erst durch diese Anforderungen kann ein Angebot erstellt werden, wobei hier oft noch Unklarheiten übrig bleiben, die zu einem späteren Zeitpunkt erst erkannt und dann erst abgestimmt werden. Werden alle Überarbeitungen vollständig abgeschlossen, kann das endgültige Angebot abgegeben werden.

Feingliederung:

Ein typischer Ablauf einer Angebotserstellung sieht in der Regel so aus:

Prinzipiell muss nach Anfrageeingang überprüft und entschieden werden, ob sich dieses Projekt lohnt bzw. ob die Wahrscheinlichkeit den Zuschlag zu erhalten, überhaupt existiert. Weiters muss geprüft werden, ob in der Firma die notwendigen Kapazitäten zur Verfügung stehen, um dieses Projekt durchführen zu können. Entschließt man sich ein Angebot zu erstellen, so muss die Anfrage auf Vollständigkeit geprüft werden und gegebenenfalls genauer spezifiziert werden. In diesem Angebotsklärungsprozess werden alle technischen Konfigurationen festgelegt, welche benötigt werden, um ein Angebot erstellen zu können. Schon vorhandene oder ähnliche Anlagen können natürlich als Grundlage verwendet werden und müssen eventuell teilweise nur noch an die neue Anlagengegebenheit angepasst werden. Detailgenaue und exakt an die Wünsche des Kunden angepasste Angebote erhöhen dabei die Chancen auf Erfolg.

Wird durch den Auftraggeber das vorgeschlagene Lösungskonzept akzeptiert, kann das Angebot erstellt und kalkuliert werden. Der Angebotspreis sollte aber auch mit einer realistischen Einschätzung des Marktes übereinstimmen, um die Chancen auf den Zuschlag zu wahren. Die erforderlichen Tätigkeiten werden zur einfachen Veranschaulichung als Flussdiagramm dargestellt: (siehe Abb. 2)

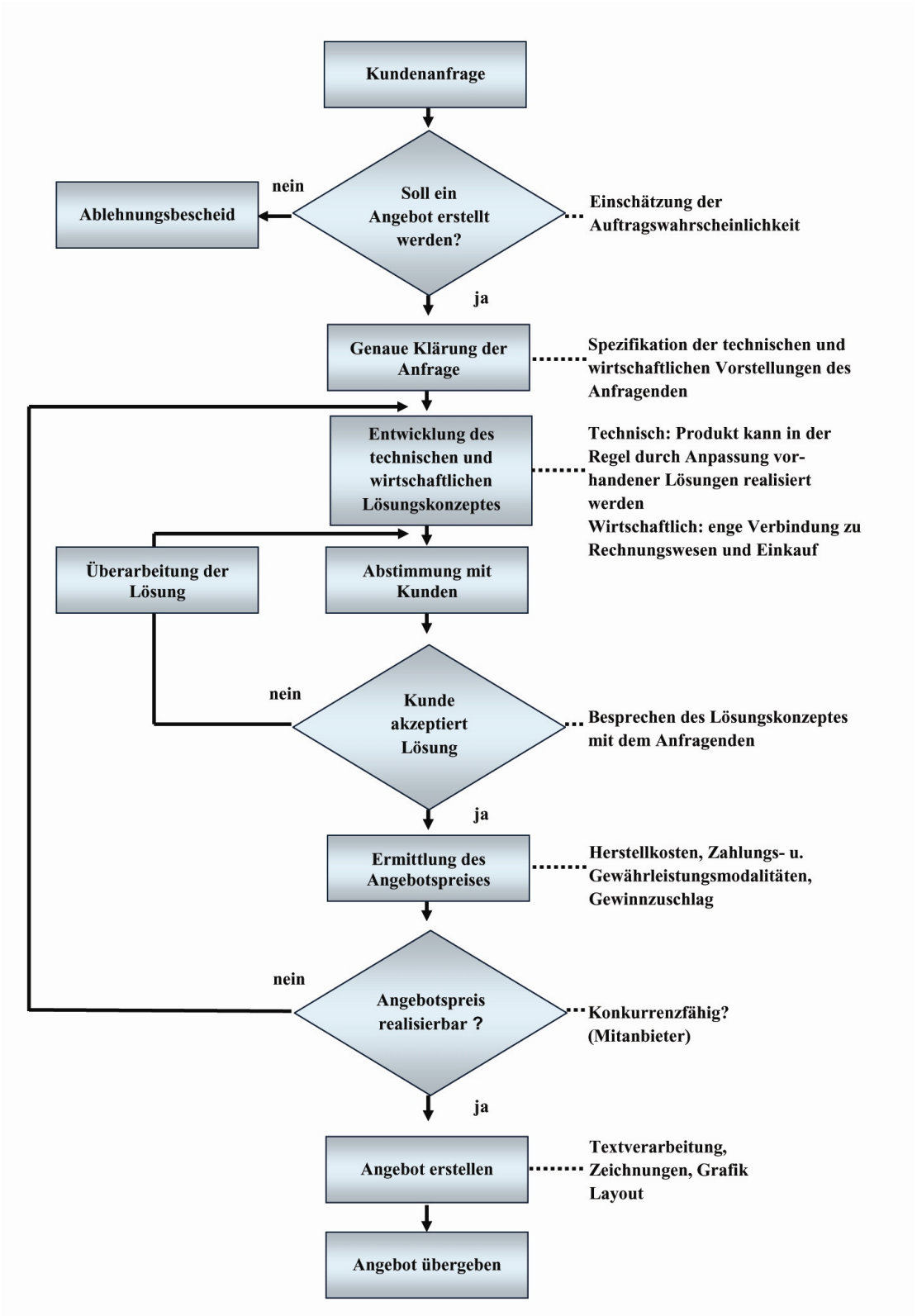


Abb. 2: Ablaufschema der Angebotserstellung[8]

[8] Richter Dipl.-Wirtsch.-Ing. R.: Stucky, Prof. Dr. rer. Nat. W: Datenbanksysteme im Angebotsbereich – Anforderungen und Trends, Karlsruhe, 1990, S181

Kalkulation:

Nachdem die Anlage im Programm projektiert wurde, erfolgt anschließend die kalkulatorische Bewertung der einzelnen Bauteile unter Zugriff auf gespeicherte Herstellungs- und Selbstkosten. Dabei finden bestimmte erschwerte Installationsbedingungen ebenso Berücksichtigung, wie kaufmännische Kriterien (z.B. auftragsgrößenabhängige und kundenspezifische Rabatte und Zahlungskonditionen). Die ermittelten Kalkulationswerte können natürlich manuell je nach Wettbewerbssituation als auch nach Marktsituation individuell im Programm modifiziert werden.[9] Dies erfolgt über eine anwenderseitig bestimmte Rabattstaffel, wobei für Geräte von unterschiedlichen Herstellern unterschiedliche Rabatte gewährt werden können. Durch diese Kalkulation steht am Ende immer die aktuelle Angebotssumme bereit.

[9] Schwetz, Dipl.Betriebsw. W.: Computerunterstützte Angebotserstellung im Rahmen integrierter Vertriebssteuerungssysteme, Pfinztal, 1990

3 Systemspezifikation

3.1 Anforderungen an das Programm

Es soll ein Programm zur Kalkulation von mess-, steuer- und regeltechnischen Anlagen erstellt werden, welches im Speziellen auf Klimatechnikregelung ausgelegt ist. Die Kalkulation soll anschließend in ein formschönes Angebot einfließen. Die gesamten Projektdaten sollen in einzelnen Dateien abgespeichert werden.

Über eine Eingabemaske soll es zunächst möglich sein, die zu steuernden klimatechnischen Geräte, nach Anlagenteilen sortiert, auszuwählen und mit den vorgegebenen Parametern zu versehen. Da es in der Praxis immer wieder Anlagenteile mit ähnlichen Aufbauten gibt, soll die Möglichkeit bestehen, die bereits fertig konfigurierten Anlagen zu kopieren und anschließend zu bearbeiten.
Um diese Funktion zu erfüllen, genügt auch ein Export in ein Excel File, wo die oben beschriebenen Bearbeitungsvorgänge durchgeführt werden können. Durch den abschließenden Import der geänderten Daten aus dem Excel File werden nun die kopierten Anlagen in das bestehende Projekt eingelesen.

3.1.1 Kalkulation Feldgeräte

Unter Feldgeräte werden alle Fühler, Sensoren, Geber, Steller und sonstige messtechnische Geräte verstanden, die zur Steuerung und Regelung vorort bei den Anlagenteilen montiert sind.
Diese Feldgeräte sind aus den eingegebenen Daten in richtiger Stückzahl herauszuzählen und nach ihren Anforderungen typenrichtig auszuwählen.

Nachdem die benötigten Feldgeräte in richtiger Type und Stückzahl vorhanden sind, soll der Gesamtpreis pro Gerätetype aufgrund des Listenpreises und der Stückzahl errechnet werden. Mit einem zentralen Verkaufsrabattsatz pro Hersteller soll der Rabatt vom Listenpreis abgezogen werden. Somit ist der Verkaufspreis der Feldgeräte kalkuliert und der Preis gegeben, welcher im abschließenden Angebot angegeben ist.

Zur Kalkulation muss der Einkaufspreis pro Feldgerätetype errechnet werden. Dieser wird über den zentralen Einkaufsrabattsatz pro Hersteller eruiert.
Es ist darauf zu achten, dass der Verkaufsrabatt nicht höher als der Einkaufsrabatt ausfällt. Dies muss durch das Programm überwacht und verriegelt sein.

Diese dazu benötigten Daten und Querverbindungen sind in der allgemeinen Datenbank vorher einzutragen. Auch für diese Datenbank gilt, dass Änderungen und Erweiterungen möglich sein müssen, da sich teilweise die Einkaufsbedingungen bei Lieferanten ändern können.

3.1.2 Kalkulation Datenpunkte

Als Datenpunkte werden die Summe aller digitalen und analogen Ein- und Ausgangssignale verstanden, welche für die digitale Steuerung und Regelung benötigt werden. (z.B.: 20 digitale Eingänge, 15 digitale Ausgänge, 4 analoge Eingänge und 3 analoge Ausgänge => Σ42 Datenpunkte)

Die in einem Projekt benötigten Datenpunkte sind aus den Daten des Projektes über die Formeln in der Datenbank herauszuzählen.

Über den Verkaufspreis für die Programmierung eines Datenpunktes, wird der Gesamtaufwand der Programmierung der Anlage errechnet.
Über den Selbstkostenpreis für die Programmierung eines Datenpunktes, werden die Herstellungskosten der Programmierung der Anlage errechnet. Diese dienen wieder als Kalkulationsgrundlage.

3.1.3 Kalkulation Regelgeräte

Unter Regelgeräten werden alle CPU's, digitale und analoge Ein- und Ausgangsmodule, Stecksockel, spezielle Trafos, Anschlussmodule, Übersteuerungsmodule und Zentralmodule verstanden, welche für die digitale Steuerung und Regelung im Schaltschrank von Nöten sind. Diese Regelgeräte sind aus der Anzahl der Datenpunkte in benötigter Stückzahl herauszurechnen.

Sind alle Regelgeräte in richtiger Type und Stückzahl bestimmt worden, so wird der Gesamtpreis pro Gerätetype aufgrund des Listenpreises und der Stückzahl errechnet. Über einen zentralen Verkaufsrabattsatz pro Hersteller soll der Rabatt abgezogen werden. Dadurch ist der Verkaufspreis der Regelgeräte kalkuliert.

Natürlich muss auch hier eine Kostenkontrolle durchgeführt werden. Dazu wird der Einkaufspreis pro Regelgerätetype errechnet und über den zentralen Einkaufsrabattsatz pro Hersteller festgelegt.
Es ist darauf zu achten, dass der Verkaufsrabatt nicht höher als der Einkaufsrabatt ausfallen darf.

In der allgemeinen Datenbank werden die Daten und Querverbindungen vorher eingetragen.

3.1.4 Kalkulation Leitwarte

Unter einer Leitwarte werden alle Computer und sonstigen Geräte verstanden, die zur Visualisierung der regeltechnischen Anlage benötigt werden.
Die Einzelteile der Leitwarte sind anhand der Gesamtzahl der Datenpunkte in entsprechender Stückzahl auszuwählen.

Die Einzelteile der Leitwarte sind nun in richtiger Type und Stückzahl vorhanden. Nun soll der Gesamtpreis pro Gerätetype aufgrund des Listenpreises und der Stückzahl errechnet werden. Über den zentralen Verkaufsrabattsatz wird der Verkaufspreis der Geräte errechnet. Somit ist der Verkaufspreis der Leitwarte kalkuliert.

Auch wird der Einkaufspreis pro Gerätetype zur Kostenkontrolle errechnet, wobei wiederum der zentrale Einkaufsrabattsatz pro Hersteller den Preis festlegt.
Es ist darauf zu achten, dass der Einkaufspreis nicht höher als der Verkaufspreis ausfällt.

Über den Verkaufspreis für die Visualisierung eines Datenpunktes, wird der Gesamtpreis der benötigten Dienstleistung für die Leitwarte errechnet.

Über den Selbstkostenpreis für die Visualisierung eines Datenpunktes, werden die Herstellungskosten der benötigten Dienstleistung für die Leitwarte errechnet.

Die dazugehörigen Daten und Querverbindungen sind in der allgemeinen Datenbank vorher einzutragen.

3.1.5 Kalkulation Schaltanlage

Unter einer Schaltanlage (auch Schaltschrank genannt) versteht man die elektrische Versorgung der Verbraucher und der elektrischen Steuerung. Dies ist die Schnittstelle zwischen Bediener und der elektrischen Anlage.

Da die Schaltanlage extern zugekauft werden muss, wird hierfür der Preis beim Schaltschrankbauer erfragt und als Einkaufpreis mit Aufschlag in das System eingegeben.

3.1.6 Kalkulation Inbetriebnahme

Als Inbetriebnahme wird die Zeit auf der Baustelle angesehen, welche zur Kontrolle und Inbetriebsetzung der Anlage sowie der Einschulung der Bediener aufgewendet wird.

Über den Verkaufspreis für die Inbetriebnahme eines Datenpunktes wird der Gesamtpreis der Inbetriebnahme der Anlage errechnet.
Über den Selbstkostenpreis für die Inbetriebnahme eines Datenpunktes werden die Herstellungskosten der Inbetriebnahme der Anlage errechnet.

3.1.7 Teilsummenaufstellung

Um den Gesamtpreis zu eruieren, werden nun die Gesamtpreise für Verkauf und Selbstkosten der oben beschriebenen Teile in eine Tabelle eingetragen und die Summen errechnet.

Das nun fertige Angebot soll mit einem formschönen Deckblatt, einem kurzen Text für jede anzubietende Gerätetype und dem gesamten Verkaufspreis ausgegeben werden.

3.2 Durchführungsplanung

3.2.1 Projektplanung

Um den Überblick auf das gesamte Projekt nicht zu verlieren, wird es in einzelne Projektteile gegliedert. Das gesamte Projekt darf dabei aber nicht aus den Augen gelassen werden. Begonnen wird mit den allgemeinen Daten für ein Projekt, wie Anlagenname, Kunde, Adresse, Datum usw. Sind alle diese Daten eingegeben, wird die Anlage genauer definiert. Handelt es sich nur um eine große Anlage oder gibt es mehrere kleinere Unteranlagen oder auch einzelne Kleinanlagen, je nach Kundenwusch. Nach der Gliederung der gesamten Anlage in kleinere Anlagenteile, werden nun die entsprechenden Teile im Leistungsverzeichnis (LV) zusammengestellt. Die erforderlichen Daten stehen dem Benutzer aus der Datenbank zur Verfügung.

Der nächste Schritt ist die Kalkulation. Dazu werden etwaige Rabatte und der Programmierstundensatz in einem eigenen Rabattblatt festgelegt. Diese Daten dienen als Grundlage der Kalkulation, um unterschiedliche Aufschläge in Angebote einrechnen zu können. Weiters wird noch der Steuerungstyp gewählt und die geschätzten Inbetriebnahmekosten inklusive der Anreise-, Abreise- und Unterbringungskosten eingetragen. Danach wird das Projekt kalkuliert und dem Benutzer werden die Preise übersichtlich angezeigt. Ist der Benutzer mit dem Angebot zufrieden, wird das gesamte Angebot inklusive aller erforderlichen Ausschreibungstexte erstellt.

Um Ergänzungen oder kleine Änderungen vornehmen zu können, kann jederzeit während der Anbotserstellung zu einem vorhergehenden Schritt zurückgekehrt werden.

3.2.2 Schnittstellen mit zusätzlich benötigten Programmen

Welche Schnittstellen werden benötigt? Welchen Programmieraufwand stellen diese zusätzlichen Programme dar und welche Funktionen werden benötigt? Alle diese Fragen müssen im Vorfeld geklärt, definiert und getestet werden. Prinzipiell kommt die Anwendung mit zwei externen Programmen in Verbindung:

a) Datenbank: Mehrere Datenbanken wurden auf Machbarkeit und benötigte Funktionen überprüft, um den Anforderungen gerecht zu werden.

b) Excel von Microsoft: Die Schnittstelle zu diesem Programm wurde vom Anwender gefordert. Daher musste diese erst entworfen und erprobt werden.

3.2.3 Absegnung der erarbeiteten Ideen durch die Firma GAA

Die erarbeiteten Überlegungen wurden mit der Firma GAA abgestimmt, damit diese Funktion in der Praxis problemlos anwendbar ist. Kleinere Überarbeitungen wurden auf Wunsch des Auftraggebers noch durchgeführt.

3.2.4 Informationsbeschaffung

Informationen zur Programmierung legten den Grundstein vor der eigentlichen Programmerstellung. Auch kleinere Testprogramme, welche gewisse gewünschte Funktionen beinhalten, wurden vorab getestet. Auch das Internet diente der Informationsbeschaffung, da es gerade im Bereich der Programmierung viele Gleichgesinnte in öffentlichen Internetforen gibt, die Hilfe zu gewissen Problemstellungen geben können.

3.2.5 Programmerstellung

Anschließend wurde mit der Herstellung des eigentlichen Programms begonnen. Durch den zuvor erstellten Projekt- und Zeitplan wurden vorab sehr viele Dinge geklärt. Die Projektplanung ist einer der wesentlichsten Punkte eines gesamten Projekts.

3.3 Zeitplan

Herbst 2007: Einarbeiten in die Programmiersoftware
Jänner 2008: Beginn der Programmerstellung
März 2009: Beginn der Diplomarbeitserstellung
Februar 2009: Fertigstellung des Programms
Bis April 2009: Praxistest mit realen Projekten
Mai 2009: Fertigstellung der Diplomarbeit mit allen erworbenen Erkenntnissen

4 Implementierung

4.1 Verwendete Programmiersoftware

Für diese Diplomarbeit wird das Entwicklungstool **Borland C++ Builder™. Enterprise Suite Version 5.0 (Build 12.34)** verwendet.

4.2 Entwicklung vom C++Builder bis heute

Borland C++ Builder 5.0 kam im Jahr 2000 auf den Markt. Es war ein revolutionierendes Tool[10], welches sich sehr weit verbreitete.

Wie aus einer Pressemitteilung[11] bekannt, wurde im Dezember 2005 eine neue Version auf den Markt gebracht: „Borland Developer Studio 2006“.

Im November 2006 wurde ein neues Unternehmen gegründet, welches in Zukunft alle Entwicklungsprodukte aus dem Hause Borland übernehmen soll. Wie in einem offenen Brief von Ben Smith (CEO, CodeGear) bekannt gegeben, wird die neue Firma CodeGear heißen.[12] Es wird die Entwicklung und Vermarktung der Produkte Developer Studio (inkl. Delphi, C++Builder und C#Builder), Turbo Delphi, Turbo C++ und Turbo C# sowie die Java-Entwicklungsumgebung JBuilder und die Datenbank InterBase übernommen.[13] Das neu gegründete Unternehmen bleibt aber zu 100% im Besitz der Firma Borland.

Im Mai 2007 kam das nun neueste Produkt von CodeGear heraus: „C++Builder® 2007“ [14]

---

[10] URL: http://dn.codegear.com/article/20781, verfügbar am 13.02.2009

[11] Barillo, Jim: URL: http://phx.corporate-ir.net/phoenix.zhtml?c=112793&p=irol-newsArticle&ID=792562&highlight=, verfügbar am 13.02.2009

[12] Smith, Ben: URL: http://dn.codegear.com/article/33818, verfügbar am 13.02.2009

[13] URL: http://entwickler.de/zonen/portale/psecom,id,214,news,32493,p,0.html, verfügbar am 13.02.2009

[14] URL: http://www.codegear.com/article/36464, verfügbar am 13.02.2009

Diese Version wurde so konzipiert, dass die kompletten Windows Vista API Funktionen und vieles mehr genutzt werden können. [15]

Diese Diplomarbeit wurde mit der Version 5.0 erstellt, da in der Firma des Autors mit diesem Entwicklungstool gearbeitet wird. Alle neu erworbenen Erkenntnisse können dadurch nützlich eingesetzt werden.

Die wichtigsten Informationen zum Anlegen eines neuen Programms:

Jedes neu angelegte Projekt erhält automatisch folgende Dateien, welche beliebig benannt werden können:[16]

- Project1.cpp: Die Quelltextdatei des Projekts.
- Unit1.cpp: Die Quelltextdatei des Hauptformulars. Diese Datei nennt man Unit.
- Unit1.h: Die Include-Datei des Hauptformulars. Diese Datei nennt man Unit-Header.
- Unit1.dfm: Eine Ressourcendatei, in der Informationen zum Hauptformular gespeichert werden. Diese Datei nennt man Formulardatei.

Jedes Formular hat eine eigene Quelltext- (Unit1.cpp), Include- (Unit1.h) und Formulardatei (Unit1.dfm). Wenn ein zweites Formular erstellt wird, werden automatisch die Dateien Unit2.cpp, Unit2.h und Unit2.dfm angelegt.

Compiler

Nach erfolgreicher Programmierung wird das Programm von einem Compiler überprüft und übersetzt. Dieser Compiler erzeugt eine Anwendungsdatei, oft auch als exe-Datei bekannt. Diese Datei ist dann auf jedem Computer lauffähig. Wichtig dabei ist, dem Borland Builder vor dem Kompilieren (übersetzen) gewisse Optionen anzugeben. Unter den Projektoptionen – Linker darf nicht „dynamische RTL verwenden“ ausgewählt sein, da es sonst auf Computern mit unterschiedlichen dll-Dateien zu Problemen kommen kann. Somit werden alle benötigen Daten, welche für ein lauffähiges Programm benötigt werden, in die exe-Datei gepackt.

[15] URL: http://www.codegear.com/products/cppbuilder, verfügbar am 13.02.2009

[16] Borland C++ Online Hilfe

Bedeutendste Komponentenbibliothek

Eine der bedeutesten Software-Komponentenbibliothek beim Borland Builder ist die Visual Component Library (VCL)[17]. Diese wird verwendet um einfaches Entwickeln von Windows-Anwendungen zu ermöglichen. Die VCL kann in den Programmiersprachen Borland Delphi, C++, C#, … verwendet werden. Die VCL basiert auf der Win32-API (Windows Application Programming Interface). Mit dieser Bibliothek wurden die meisten Objekte der neuen Applikation realisiert. Genaueres siehe Kapitel 5.

4.3 Hardwarevoraussetzung

Um das Programm auch ohne Probleme nützen zu können, gibt es eine Mindestanforderung an den Computer:

- Betriebssystem: Windows 2000, XP, Vista
- Freier Festplattenspeicher: mindestens 5MB
- Handelsübliche Grafikkarte

An Arbeitsspeicher und CPU werden keine besonderen Ansprüche gestellt. Folgendes wird aber empfohlen:

- CPU: Pentium 4, 1.6 GHz (oder entsprechender Athlon Prozessor)
- Mindestens 512MB Arbeitsspeicher

Eine optimale Programmvisualisierung erhält man ab einer Auflösung von 1400x1050 Pixel.

4.4 Kurze Programmvorstellung

Mit diesem Programm soll ein leichtes und einfaches Arbeiten möglich sein. Kalkulationen und Angebote können ohne viel Mühe und durch einfache Konfiguration der kompletten Anlage erstellt werden. Durch Vorlage eines schon bestehenden Projektes wird dem Benutzer diese Eingabe sehr vereinfacht.

Über die Eingabe wird dem Benutzer aus einer Datenbank das entsprechende Angebot erstellt. Um die Datenbank zu aktualisieren gibt es eine Verwaltungsmöglichkeit.

---

[17] Schumann, Hans-Georg: C++ leicht & verständlich, 2. Aufl., Knowware, 2007

In dieser Datenbank sind alle möglichen Geräte und Informationen gespeichert.
Die Kalkulation wird übersichtlich zusammengefasst, wobei man jederzeit einen beliebigen Teil des Angebotes detailliert abrufen kann. Über ein Rabattblatt kann für bestimmte Kunden ein nochmals überarbeitetes Angebot gelegt werden.
Die Ausgabe erfolgt über das Microsoftprogramm „Excel“, wo das komplette Angebot zur Archivierung in einer Exceltabelle gespeichert wird.
Alle in das Projekt eingegebenen Daten werden in einer Datei gespeichert. Dies ermöglicht einen jederzeitigen Aufruf eines schon abgeschlossenen Projekts.

4.5 Vorgehensweise der Programmerstellung

In diesem Abschnitt werden die wesentlichsten Teile des Programmcodes vorgestellt und erläutert. Es wird mit Beispielen aus dem Programm gezeigt, wie diese realisiert wurden.

4.5.1 Prozeduren und Funktionen

Prozeduren und Funktionen, im C++Builder spricht man von Methoden, in ein Programm eingebaut, bringen nicht nur eine bessere Übersicht ins Programm, sondern es können immer wiederkehrende gleiche Bearbeitungsschritte zusammengefasst werden.[18]

Prozedur
Hier werden Abarbeitungsschritte zusammengefasst, welche öfters im Programm verwendet werden. Dadurch erspart man sich die Programmierarbeit und das Programm wird dadurch übersichtlicher. Um eine Prozedur verwenden zu können, muss diese zuerst in der „Headerdatei“ deklariert werden. Die normale Quelltextdatei – oft auch als UNIT bezeichnet hat die .CPP Kennung, wobei die Headerdatei eine .H Kennung hat.
Bsp.:
In dieser Anwendung wurde eine Prozedur „*Lesen*“ vereinbart, welche die Aufgabe hat eine vorgegebene Datei einzulesen.

[18] Goll Joachim: Dausmann, Manfred: Bröckl, Ulrich: C als erste Programmiersprache, 5. Aufl. Teubner, Wiesbaden, 2005

Vereinbarung in der Headerdatei:

```
private:        // User declarations
void Lesen(AnsiString);
```

In der UNIT kann nun diese Prozedur mit Anweisungen befüllt werden, um danach im Programm verwendet werden zu können. Diese Anweisungen dieser Prozedur sehen in der Anwendung beispielsweise so aus:

```
void TF_Main::Lesen(AnsiString PrjFile)
{
TProgressBar *p = new TProgressBar(StatusBar1);
p->Parent = StatusBar1;
p->Top = 2;
p->Left = StatusBar1->Panels->Items[0]->Width + 2;
        :
StatusBar1->Repaint();

 TStringList* myList = new TStringList();
 myList->LoadFromFile(PrjFile);
        :
delete myList;
 delete p;
}
```

Über eine Stringvariable (PrjFile) wird die vorgegebene Datei in die Prozedur übertragen. Danach werden eine Fortschrittsanzeige und die eigentlich gewünschten Anweisungen, das Auslesen der Datei durchgeführt.

Der Aufruf in dem normalen Quelltext ist so realisiert:

```
Lesen(OpenDialog1->FileName);
```

Funktionen

Verhalten sich im Prinzip genauso wie eine Prozedur, nur wird auch ein Wert als Ergebnis zurückgegeben. Eignet sich besonders gut um jegliche Berechnungen durchführen zu können.

Bsp.:

Die Deklaration in der Headerdatei sieht in diesem Beispiel so aus:

```
// Berechnung der Rabattprozente
float __fastcall Rabkalk(AnsiString RabGr);
```

Die Funktion *Rabkalk* hat in diesem Fall die Aufgabe die Prozente der entsprechenden Rabattgruppe, welche über Buchstaben definiert ist, aus einer Tabelle auszulesen. Dabei wird über den String *RabGr* der entsprechende Buchstabe in die Funktion übergeben. Über *return (Prozent)* werden die Prozente als Ergebnis zum Aufruf zurückgeliefert.

```
float __fastcall TF_Main::Rabkalk(AnsiString RabGr)
{
float Prozent = 1;
int comp=0;
for (int i=0;i<=StringGrid_VK->RowCount;i++)
 {
 comp = AnsiCompareStr(RabGr, StringGrid_VK->Cells[0][i]);
 if (comp == 0)
   {
    Prozent = 1-StrToFloat(StringGrid_VK->Cells[1][i])/100;
    break;
   }
 }
return (Prozent);
}
```

Aufruf dieser Funktion:

```
Summe_List1=Summe_List1+(StrToFloat(StringGrid_GLT->Cells[9][i])
*StrToInt(StringGrid_GLT->Cells[1][i])*Rabkalk(StringGrid_GLT->Cells[8][i]));
```

Der Preis eines Gerätes multipliziert mit der Stückzahl und der Rabattprozente wird der Summe hinzugefügt. Um die Prozente zu erhalten, wird die Funktion *Rabkalk* verwendet. Da die Berechnung für jedes Gerät erfolgt, ist diese Funktion nicht wegzudenken.

4.5.2 Klassen

Gehören zu den Kriterien die entscheiden, ob eine Programmiersprache als objektorientiert bezeichnet werden kann. Sie ermöglichen das Zusammenfassen von Daten und Funktionen. Damit kann verdeutlicht werden, dass diese Daten und Funktionen zusammen gehören. Es werden in der Klasse notwendige Eigenschaften und zur Manipulation der Eigenschaften notwendige Methoden zusammengefasst.[19] Es können aber auch Klassen von anderen Klassen

[19] Schumann, Hans-Georg: C++ leicht & verständlich, 2. Aufl., Knowware, 2007

abgeleitet werden. Ist dies der Fall, so spricht man von Vererbung, wobei immer einige oder alle Eigenschaften und Methoden von der vererbenden Klasse (Basisklasse) vererbt werden. Klassen werden im C++Builder immer mit einem vorangestelltem „T“ gekennzeichnet. Die Vereinbarung der Klasse wird in der Headerdatei eingetragen.

Bsp.:

```
class TKalkulation : public TForm
{
__published:   // Von der IDE verwaltete Komponenten
        TPageControl *PageControl1;
        TTabSheet *TabSheet_FG;
        TTabSheet *TabSheet_DDC;
        TTabSheet *TabSheet_SS;
        TTabSheet *TabSheet_SSBG;
            :
public:
        void ClearKalk(void);
            :
};
```

Die Definition der Methoden einer Klasse wird in der Quelltextdatei erstellt. In unserem Beispiel wird die Methode *ClearKalk* so definiert:

```
void TKalkulation::ClearKalk(void)
{
//löschen StringGrid_FuG
for (int i = 1 ; i <=StringGrid_FuG->RowCount; i++)
  for (int j = 0 ; j <=StringGrid_FuG->ColCount; j++)
      StringGrid_FuG->Cells[j][i] = "";
            :
}
```

Wird diese Methode aufgerufen, so werden Inhalte bestimmter Objekte gelöscht und auf Null gesetzt.

Nun wurde die Klasse deklariert und definiert. Um sie aber nun auch verwenden zu können, muss zu aller erst ein Objekt erzeugt werden. In dieser Anwendung wird jeweils eine Variable für jedes Bauteil vereinbart. Durch den Stern in der Vereinbarung wird signalisiert, dass es sich nicht um eine Variable sondern um einen Zeiger handelt, welcher auf eine Adresse im Arbeitsspeicher zeigt. Zeiger haben unter anderem den Vorteil, gewisse Operationen zu vereinfachen und zu beschleunigen. Jedoch muss auch auf die richtige Verwendung geachtet werden, da es leicht zu Fehlern kommen kann.

Bsp.:

*TKalkulation *pKalkulation_Baut1, *pKalkulation_Baut2;*

Über new wird der entsprechend benötigte Speicherbereich für diese Klasse freigegeben.

```
pKalkulation_Baut1 = new TKalkulation(TabSheet_Baut1);
pKalkulation_Baut1->Parent = TabSheet_Baut1;
```

Mit diesem Zeiger werden im Quelltext nun die gewünschten Methoden oder Eigenschaften verwendet. Hier das Beispiel über den Aufruf der Methode *ClearKalk* :

```
//löschen der Kalkulation
pKalkulation_Baut1->ClearKalk();
```

Neben der vereinfachten Programmierung, da für jedes Bauteil dieselbe Klasse definiert wurde, wurde auch der Quelltext stark strukturiert.

4.5.3 Datensicherung

Variablen sind nur während der Laufzeit des Programms aktiv, können aber keine Daten nach Beendigung des Programms speichern. Um später auf vorhandene, in der Vergangenheit erstellte Projekte zugreifen zu können wird die Datensicherung der Projekte auf einem externen Datenträger gespeichert. Dazu gibt es unterschiedliche Möglichkeiten wie:

- Magnetplatten (Festplatten oder Disketten)
- Magnetbänder
- Optische Platten, darunter versteht man auswechselbare Massenspeicher, welche mittels einer optischen Abtastung beschrieben und gelesen werden können, wie z.B. DVD, CD-Rom, …Die optische Abtastung erfolgt meist über einen Laser.
- Elektronische Speicherung: USB-Massenspeicher (*USB-Speicher-Stick en: USB Flash Drive)*
- Papier

In diesem Projekt ist die Sicherung mittels Dateien auf Festplatten vorgesehen. Um dies zu realisieren, muss zuerst eine Stream-Variable mit den Klassen *fstream, ifstream* oder *ofstream* definiert werden. Dann ist es möglich eine Datei anzusprechen. Im C++-Programm stehen diese Klassen mit dem Include:

#include <fstream>
using namespace std;
dem Programmierer zur Verfügung.

ifstream: Diese Klasse wird nur zum Lesen einer Datei verwendet, da sie keine Elementfunktionen zum Schreiben hat. Über das Argument *mode* wird festgelegt, welche Operation mit der Datei durchgeführt wird. In diesem Fall ist das Argument immer *ios::in*.
ofstream: Diese Klasse wird nur zum Schreiben einer Datei verwendet, da sie keine Elementfunktionen zum Lesen hat. Das Argument für *mode* ist immer *ios::out*.
fstream: Diese Klasse wird zum Schreiben und zum Lesen einer Datei verwendet.
Hier muss über das Argument *mode* festgelegt werden, welche Operation durchgeführt werden soll. Wird dies verabsäumt, so wird ein Fehler ausgegeben.

fstream f; //Variablendefinition
if (OpenDialog1->Execute())
{
f.open(OpenDialog1->FileName.c_str(),ios::binary|ios::app);
//öffnet die Datei aus dem Open Dialog1
//binary: eine Binärdatei wird angelegt
//app: Falls diese Datei vorhanden ist, wird sie geöffnet, wenn nicht, wird sie erstellt.

Aufbau des Streams:
Ein *Stream* ist ähnlich wie ein Magnetband aufgebaut. Die Daten liegen in einer Folge von Datensätzen in der Datei. Dabei gibt es einen Positionszeiger, welcher auf eine bestimmte Position in dieser Datei zeigt. Um Daten zu schreiben oder zu lesen, muss zuerst der Positionszeiger auf den richtigen Datensatz positioniert werden. Danach ist es erst möglich die gewünschten Daten zu erreichen.
Dieser Aufbau birgt natürlich einen gewissen Vorteil gegenüber Arrays, da z.B. die Größe des Files beim Entwurf nicht festgelegt werden muss.
Nachteilig ist zu erwähnen, dass bei einem Array ein bestimmtes Element direkt angesprochen werden kann, was bei einem *Stream* nicht so einfach möglich ist.

Wird eine Datei geöffnet oder bearbeitet, kann es sein, dass ein Fehler auftritt. Dies kann einfach über diese Funktion abgefragt werden.

```
if (f); // kein Fehler
else {ShowMessage("Fehler beim Öffnen");
    void clear(); //löschen des Fehlerbits
    }
}
```

Um nun Daten in eine Datei zu schreiben, wird die Elementfunktion *write* der Stream-Klasse fstream verwendet. Hier werden Daten in der Größe eines Integerfeldes ab der Adresse i in die Datei geschrieben. Die Daten müssen den Datentyp *char* aufweisen. Kommt ein anderer Datentyp zum Einsatz, so muss über die Typkonversion (char*) der Datentyp *char* hergestellt werden.

Bsp.: *f.write((char*)&i,sizeof(int));*

Nach einer Schreibfunktion soll unbedingt getestet werden, ob diese auch erfolgreich war:
if (!f) ShowMessage("Fehler bei write");

Über den Aufruf *write* werden die Daten in die Datei geschrieben. Dies erfolgt aber über das Betriebssystem, d.h. das Betriebssystem schreibt die Daten vorerst in einen Zwischenspeicher und erst, wenn der Speicher voll ist, werden die Daten in die externe Datei geschrieben. Das hat zwar auf das Programm keine Auswirkungen, aber sollte das Programm abstürzen, bevor die Daten in die Datei geschrieben wurden, so sind diese verloren.
Um dieses Problem zu umgehen, gibt es die Funktion *flush*, womit die Daten aus dem Zwischenspeicher in die Datei geschrieben werden. Die beste Sicherstellung bekommt man, indem man jedes mal nach einer *write* Opteration die Funktion *flush* aufruft. Zwar wirkt sich dies auf die Bearbeitungsgeschwindigkeit aus, aber dafür ist das Risiko des Datenverlusts geringer.

Bsp.: *ostream& flush();*

Um nun eine Datei zu lesen, muss sie zuvor geöffnet werden. Nach dem Öffnen einer Datei steht der Positionszeiger immer am Anfang der Datei. Sollte die Datei schon offen sein, so muss zuerst der Positionszeiger an die richtige Stelle der Datei geschoben werden.
Die Funktion *read* wird nun zum Auslesen verwendet. Hier werden Daten aus der Datei in der Größe eines Integerfeldes in den Speicherbereich ab der Adresse i gelesen.
Auch diese Daten müssen den Datentyp *char* aufweisen. Kommt ein andere Datentyp zum Einsatz, so muss über die Typkonversion (char*) der Datentyp *char* hergestellt werden.

Bsp.: *f.read((char*)&i,sizeof(int));*

Tritt während des Lesens ein Fehler auf, so sollte auch dies dem Benutzer gemeldet werden:
Bsp.*: if (!f.eof()) ShowMessage("Fehler bei read");*
else ; // f.eof(): kein Fehler

Eine nützliche Funktion, eine gesamte Datei auszulesen, ist die *eof().* Der Zustand der Stream-Variablen zeigt an, wenn über das Ende der Datei gelesen wird.
bool eof();

Die Verbindung zwischen einer Stream-Variablen und einer Datei wird über die Elementfunktion
void close();
getrennt. Es werden nicht gespeicherte Daten aus Zwischenpuffern in die Datei geschrieben und eventuell vorhandene Reservierungen der Datei beim Betriebssystem aufgehoben.

LoadFromFile/SaveToFile

Diese beiden Funktionen sind eine weitere Möglichkeit, welche der C++ Builder zum Speichern und Laden von Dateien bietet. Diese Funktionen gehören zu der Basisklasse TStrings, welche für die Benutzung von Stringkomponenten verwendet werden. Über eine nicht visuelle Stringliste der Komponente TStringList wird das Lesen, Speichern und Verwalten vereinfacht.

Mit der Methode *LoadFromFile* wird eine Stringliste aus einer Datei geladen. Ein Parameter wird dabei übergeben, welcher den Namen der zu ladenden Textdatei enthält. Daraufhin werden die einzelnen Zeilen der Textdatei jeweils in einen eigenen String eingelesen.

Mit der Methode *SaveToFile* wird eine Stringliste in eine Datei gespeichert. Ein Parameter wird dabei übergeben, welcher den Namen der zu speichernden Textdatei enthält. Jeder String wird in eine neue Zeile geschrieben. Ist die Datei vorhanden, wird sie überschrieben, falls nicht, wird sie neu angelegt.

Stringlisten bieten mit ihren Methoden und Eigenschaften besondere Vorteile, um Strings leichter verwalten zu können:[20]

- Mit Hilfe der Methoden *Add()* oder *Insert()* können Strings leicht an einer beliebigen Stelle der Liste eingefügt werden
- Über die Eigenschaft Strings[int index] kann jeder einzelne String der Liste leicht über seine Indexposition ausgelesen werden
- Über die Methoden *Move()* und *Exchange()* können Positionen der Strings in der Liste verändert werden
- Weiters kann die Position eines Strings über *IndexOf()* oder *Find()* in der Liste ermittelt werden
- Um eine Liste sortieren zu können, gibt es die Methode *Sort()*

Bsp.:

```
TStringList* myList = new TStringList();
myList->LoadFromFile("D:\\BorlandC++\\Projekte\\DI_Arbeit_2\\captions_gb.txt");
MainMenu1->Items->Items[0]->Caption = myList->Strings[0].SubString(9,15);
MainMenu1->Items->Items[1]->Caption = myList->Strings[10].SubString(9,15);
MainMenu1->Items->Items[2]->Caption = myList->Strings[20].SubString(9,15);
MainMenu1->Items->Items[3]->Caption = myList->Strings[30].SubString(9,15);
delete myList;
```

[20] ProBiq, Borland C++ Builder 5.0 Lehrbuch

Hier wird die Textdatei „captions_gb.txt“ in die StringList „myList“ eingelesen. Bestimmte Strings werden dann an die Komponente *MainMenu1* als Caption übergeben.

Da das Programm in jeden beliebigen Ordner installiert werden kann, ist eine fixe Verzeichnisangabe nicht sinnvoll. Dazu eignet sich die Funktion *GetCurrntDir().* Damit kann man das aktuelle Verzeichnis auslesen und somit ist dieses Programm in jeden beliebigen Ordner installierbar.

```
AnsiString Dir;
Dir = GetCurrentDir();  //liefert aktuelles Verzeichnis
```

4.5.4 Fehler – „Bugs“ und Exceptions

Fehler, oft auch *„Bugs“* genannt, sind ein großes Kapitel in der Softwareprogrammierung. In jedem noch so kleinen Programm kann sich ein kleiner, oft auch nicht wirklich auf die Funktion auswirkender Fehler einschleichen. Ein Programm zu erstellen, welches keine Fehler beinhaltet, ist so gut wie unmöglich. Je nachdem, wie wichtig eine Software ist, schätzt man, dass sich in 1000 Zeilen Programmcode etwa zwei bis fünfundzwanzig Fehler befinden.[21]

Es gibt natürlich viele Fehler, welche bei einem Programm auftreten können. So gibt es z.B. Syntaxfehler und Laufzeitfehler, um die zwei wichtigsten Fehler aufzuzählen.

- Syntaxfehler: sind solche, welche während des Programmierens durch Tippfehler oder falsch geschriebener Syntax entstehen. Der Compiler überprüft vor dem Programmstart das gesamte Programm, um solche Fehler zu suchen. Wird ein Fehler gefunden, wird dem Programmierer die Stelle des Fehlers mit einer Fehlermeldung angezeigt. Es ist aber nicht möglich, das Programm ohne Behebung dieser Bugs zu starten.
- Laufzeitfehler: sind besonders tückisch, da sie erst während der Laufzeit, also während der Programmausführung auftreten können. Ursachen dafür können falsche Implementierung von Funktionalitäten, falsche Laufzeitumgebungen (wie falsche

[21] URL: http://www-aix.gsi.de/~giese/swr/ursache1.html, verfügbar am 01.03.2009

Betriebssystemversion), fehlerhafter Speicherzugriff, usw. sein. Zwar gibt es schon viele Programmiertools, welche gewisse Fehler nicht zulassen, (z.B. durch automatische Speicherbereinigung wie bei Java) aber trotzdem muss der Programmierer sehr sorgfältig bei der Programmerstellung vorgehen. Würde das fertige Programm alleine auf einem Rechner laufen, so würde es bei einem Laufzeitfehler abstürzen oder nicht mehr auf den Benutzer reagieren. Der Programmierer kann in der C++Builder Entwicklungsumgebung bei Auftritt solch eines Fehlers das laufende Programm abbrechen, den Fehler ausfindig machen und beheben. Es gibt aber auch die Möglichkeit das Programm auf gewisse Fehler quasi vorzubereiten.

Unter Exceptions (Ausnahmen bzw. Ausnahmesituationen) versteht man aufgetretene Programmzustände, welche meist Fehlerzustände sind. Kann zum Beispiel einer Speicheranforderung nicht stattgegeben werden, oder soll eine Datei geöffnet werden, welche aber durch ein anderes Programm gerade verwendet wird, wird eine Exception ausgelöst. Das Programm kann nun einen definierten Algorithmus abarbeiten, um den Fehler zu ignorieren, zu beheben oder anzuzeigen. Richtige Anwendungen von Ausnahmen ergeben ein stabiles Programm, welches auf unvorhergesehene Probleme flexibel reagiert.
Um einen Programmabschnitt auf Exceptions überprüfen zu lassen, wird beispielsweise folgende Anweisung verwendet:

```
try
 {
  Excel = GetActiveOleObject("Excel.Application");
 }
catch(...)
 {
  Excel = CreateOleObject("Excel.Application");
 }
```

In diesem Fall wird ein aktives Excel-Programm für die Schnittstelle bereitgestellt. Läuft aber keine Excel Anwendung, so muss das Programm Excel zuerst gestartet werden, um es dann für die weitere Verwendung vorbereiten zu können. D.h. tritt während der Abarbeitung des *try*-Blocks eine Exception auf, so wird sofort in den *catch*-Zweig verwiesen. Dort wird der Algorithmus für das Starten der Excelanwendung eingetragen und bei einer Exception

ausgeführt. Hätte man dies nicht so programmiert, würde während der Laufzeit das Programm mit einem Fehler abbrechen.

Natürlich gibt es auch andere Fehler wie Konzeptfehler, logische Fehler und auch Bedienungsfehler, welche unbedingt vermieden werden müssen.

Reproduzierbarkeit von Programmfehlern ist ein wichtiger Punkt um Fehler genau zu lokalisieren und beheben zu können. Oft passiert es, dass bei einem Wiederholungsversuch es nicht mehr möglich ist, diesen Fehler zu produzieren. Dies hat verschiedene Ursachen. Es kann vorkommen, dass der durch die Aktivierung des Fehlers hervorgerufene Programmabsturz etwas verzögert wird. Dadurch wird die Erkennung sehr erschwert. Aber auch andere Elemente des Systems können mit dem Programm in Konflikt geraten, wenn der Programmierer nicht richtig mit den vorhandenen Ressourcen umgeht.

Programmierarbeit muss sehr genau und sorgfältig durchgeführt werden. Es gibt aber bereits sehr viele Hilfsmittel, um gewisse Fehler zu vermeiden. Trotz allem werden noch immer genügend Fehler gemacht. Ein genauer Test des Programms ist daher immer erforderlich.

4.5.5 Datenbanksystem

4.5.5.1 Allgemeines

Prinzipiell wird das Datenbanksystem in 2 Teile gegliedert – das Datenbankmanagementsystem (DBMS) und die Datenbank (DB) selbst. Das DBMS beinhaltet die Verwaltungssoftware, welche für die interne Speicherstrukturierung zuständig ist und welche alle Lese- und Schreibzugriffe auf die Datenbank kontrolliert. Die Menge der physisch abgespeicherten Daten wird als Datenbank bezeichnet.

4.5.5.2 Datenunabhängigkeit

Vorteil eines Datenbanksystems ist unter anderem die physische Datenunabhängigkeit. Anwendungsprogramme (AP) greifen dabei nicht direkt auf die Datei zu, sondern erhalten die Daten von dem DBMS. Dieses stellt dem jeweiligen Programm die benötigten Daten in einer „logischen“ Datei zur Verfügung, wobei das DBMS darauf zu achten hat, dass die logischen Daten abbildmäßig mit den physischen Daten in Zusammenhang stehen. Dieses Konzept wird in folgender Abbildung veranschaulicht:

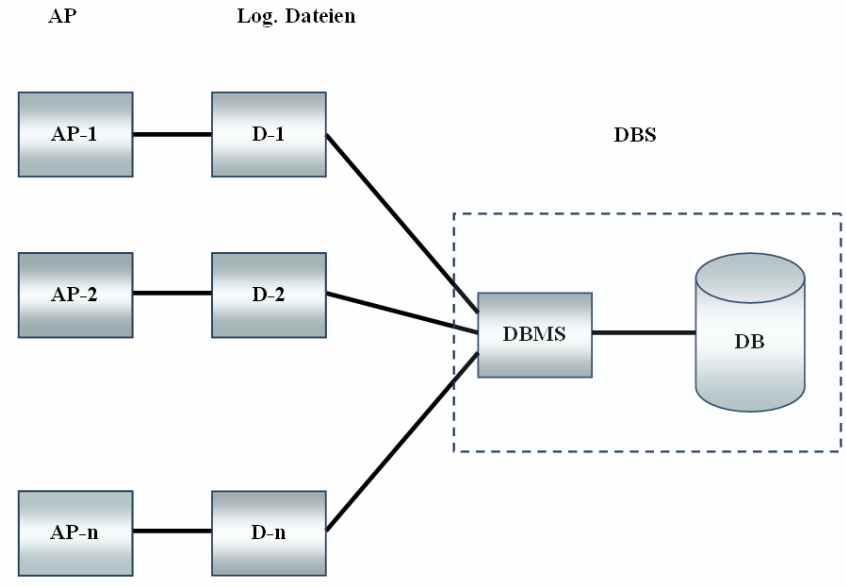


Abb. 3: Konzept eines DBMS[22]

[22]Corsten, H: Software-Architekturen für das E-Business, 2000, S. 177.

4.5.5.3 Datenredundanz

Einen weiteren positiven Einfluss eines Datenbanksystems bildet die Vermeidung von Datenredundanz. Das DBMS hat Sorge zu tragen, dass sich die aktualisierten Daten in den logischen Dateien auf die physischen Daten auswirken. Zugleich natürlich müssen alle anderen Anwendungsprogramme, welche gleichzeitig auf diese Daten zugreifen, eine neue logische Datei mit neuen Inhalten erhalten. Da dies sofort erfolgt, kann es nicht passieren, dass es unterschiedliche Zustände gibt, d.h. es kann keine Datenredundanz entstehen.

4.5.5.4 Mehrbenutzerfähigkeit

Mehrbenutzerfähigkeit und flexibler Gebrauch von Daten ist ein weiteres Feature, welches durch ein Datenbanksystem gegeben ist. Das DBMS stellt dem Anwendungsprogramm schon bestimmte häufig verwendete Operationen zur Verfügung, um den Datenzugriff zu vereinfachen. Diese müssen dann im Anwendungsprogramm nicht extra programmiert werden. Auch Kombinationen von Operationen können realisiert werden (siehe unten).
Die Konsistenz von Daten ist immer gegeben, da es durch die zentrale Datenhaltung für gleiche Daten keine unterschiedlichen Werte geben kann.

Zusammengefasst hat ein Datenbanksystem folgende Vorteile:

- Einfache Suche in großen Datenbeständen
- Mehrere Anwender können auf denselben Datenbestand zugreifen (ohne dass sich Daten überschreiben)
- Flexibler Umgang mit Daten
- Keine Datenredundanz
- Mehr Komfort
- Bessere Effizienz der Datenhaltung
- Einfache Verwendung von Sicherheitsfunktionen

Datenbanken werden heutzutage überall eingesetzt, wo es eine größere Ansammlung an Daten gibt. Sei es in Firmen, wo alle Geräte zusammengefasst sind oder alle in einem

Unternehmen verwendeten Bauteile, oder Messwerte, welche über ein bestimmtes Ereignis aufgezeichnet worden sind, usw.; man spricht dann von Gerätedatenbanken, Bauteildatenbanken, Messwertdatenbanken, usw. Alle diese Daten sind in entsprechende Datensätze gegliedert und es ist sehr einfach mit diesen Datenbanken zu arbeiten.

Heutzutage gibt es bereits eine Unzahl von verschiedenen Datenbanksystemen.
Die Datenbanksysteme, mit welchen die VCL (Visual Component Library) arbeiten kann, sind:[23]

- dbGo: der Zugriff auf diese Datenbanken erfolgt über die ADO-Schnittstelle (Microsoft ActiveX Data Objects). Sehr verbreitete Datenbanken, wie Microsoft Access, SQL-Server (Structured Query Language), Informix, Oracle-Datenbanken und Datenbanken mit ODBC oder OLEDB-Treiber gehören zu dieser Kategorie.
- BDE (Borland Database Engine): dazu gehören Datenbanken wie Paradox, dBASE, FoxPro und CSV-Format (kommagetrennte ASCII-Werte). Alle diese DB sind für einfache Anwendungen sehr gut geeignet, da sie keine Installation eines Datenbankservers voraussetzen. Dafür haben sie aber nur einen beschränkten Funktionsumfang im Vergleich zur SQL-Datenbank
- InterBase: für Interbase Datenbanken
- dbExpress: für SQL-Datenbanken

Die Darstellung von Datenbanken kann man sich vereinfacht als Tabellenform vorstellen. D.h. ein Datensatz entspricht einer Zeile, wobei pro Spalte Werte mit unterschiedlichen Datentypen vorhanden sein können. Alle diesen Daten weisen dem Datensatz (Zeile) verschiedene Werte zu.

Bsp: Girokontodatenbank

Datensätze (ID)	Nachname	Vorname	Geburtsdatum	Kontostand
1	Huber	Herbert	23.11.1976	3.143,04
2	Maier	Johann	12.05.1981	24.128,33

Tab. 2: Beispiel einer simplen Datenbank

[23] Kaiser, Prof. Richard, C++ mit dem Borland C++Builder 2007, 2. Auflage, Tübingen, 2008

DataSource Komponente:
Die Tabelle kann mit der Komponente DBGrid angezeigt werden. Mit Hilfe des DBGridNavigator können Datensätze gelöscht, geändert oder neu hinzugefügt werden. Um diese Komponenten auch verwenden zu können, muss die Tabelle über eine DataSource Komponente mit dem DBGrid und dem DBNavigator verbunden werden. Dadurch erhält man eine einfache Möglichkeit die Datenbank zu administrieren.
Um jedoch mit der Datenbank auch laufzeitabhängig arbeiten zu können, gibt es SQL-Abfragen.

SQL-Abfragen: (Structured Query Language – strukturierte Abfragesprache)
Diese Sprache hat sich im Laufe der Zeit als Standard für Datenbankzugriffe durchgesetzt. Sie ist eine sehr umfassende Sprache mit der Abfragen und Anweisungen zum Erstellen oder zum Ändern möglich sind.

Eine SQL-Anweisung wird als Text zusammengestellt und dann durch ein „Open“ oder „ExecSQL“ ausgeführt. Vor einer neuen Anweisung muss ein „Close“ aufgerufen werden.

Befehle zum Abrufen von Daten

SELECT * ….....	bewirkt, dass alle Spalten der Tabelle angezeigt werden. Um nur eine bestimmte Spalte anzeigen zu lassen, muss nach SELECT der Spaltenname angegeben werden.
FROM x ……	x repräsentiert den Datenbanknamen, mit welcher Datenbank gearbeitet werden soll.
WHERE x ...…..	Hiermit werden alle Datensätze angezeigt, welche diese Bedingung erfüllen. Alle anderen Datensätze aus der Datenbank werden nicht berücksichtigt.
ORDER BY x….	alle ausgewählten Datensätze werden nach der Spalte „x“ sortiert.
GROUP BY x …	gruppiert alle Zeilen nach der Spalte „x“.
LIKE `%x%` ….	im Zusammenhang mit anderen Kommandos kann hier die Auswahl auf alles beschränkt werden, wo ein „x“ vorkommt.

Befehle zur Datenmanipulation

INSERT ……..... Hinzufügen von einem Datensatz

UPDATE …….... Ändern von Datensätzen

DELETE …...… Löschen von Datensätzen

Bsp.:

```
ComboBox_DDC->Clear();
Table1->Open();
Query1->Close();
Query1->SQL->Clear();
AnsiString ASSQLCommand = "SELECT * FROM Angebot_DB2_Aktuell WHERE Bezeichnung LIKE '%CPU%' ORDER BY ID";
Query1->SQL->Add(ASSQLCommand);
Query1->Open();
Query1->Active = true;
Query1->First();
//ComboBox_DDC befüllen
while (!Query1->Eof)
 {
  ComboBox_DDC->Items->Add(Query1->FieldByName("Bezeichnung")->AsString);
  Query1->Next();
 }
```

In diesem Beispiel wird in der Anwendung auf eine Datenbank „Angebot_DB2_Aktuell“ zugegriffen. Der Zugriff erfolgt über ein SQL Kommando, welches einem Query zugewiesen wird. Durch den Aufruf von *Open* oder *ExecSQL* wird die Anweisung ausgeführt. Nun werden alle Einträge aus der Datenbank abgefragt, in welchen das Wort „CPU“ in deren Bezeichnung vorkommt. Weiters werden die abgerufenen Datensätze nach dem Datenfeld „ID“ sortiert. Die Datensätze werden dann in eine ComboBox dem Benutzer als Auswahl zur Verfügung gestellt.

4.5.6 Excel

Die Ausgabe soll in einer Exceltabelle erfolgen. Dies ermöglicht eine übersichtliche Darstellung und sie kann auf jedem anderen Computer angesehen werden.

Um eine Kommunikation zu dem Excelprogramm aufbauen zu können, wird die COM – Schnittstelle (Component Object Model) verwendet. Dieses sprachunabhängige Modell sorgt

für Kommunikation zwischen Anwendungen auf Windows-Plattformen und dafür, dass zwischen Komponenten und Anwendungen und zwischen Clients und Servern eine klar definierte Schnittstelle vorliegt.
Diese Schnittstelle stellt eine Spezifikation und eine Implementierung dar. Darin ist festgelegt, wie Objekte erzeugt werden und wie sie miteinander kommunizieren. Durch diese Festlegung ist eine flexible Anwendung auf unterschiedlichen Plattformen und in unterschiedlichen Prozessräumen möglich.
In der COM-Bibliothek werden eine Reihe von Diensten zur Verfügung gestellt, wie z.B. API-Funktionen, mit deren Hilfe COM-Objekte erstellt und verwaltet werden können. Im C++Builder werden über VCL-Objekte auf solche COM-Schnittstellen zugegriffen.
Die COM Schnittstelle wurde im Laufe der Zeit immer mehr erweitert und dient nun als Basis für andere Technologien, wie z.B. Automatisierung, ActiveX-Steuerelemente, Active-Dokumente und Active-Verzeichnisse.[24]

Vorteile in der Einbindung des Excel-Com-Servers in das Programm:

- Leichte Editierbarkeit von Daten (wie z.B. leichte Anpassung von Preisänderungen, Typenänderungen usw.)
- Leichtes Löschen von Datensätzen (wie alte, nicht mehr im Vertrieb befindliche Produkte)
- Leichtes Hinzufügen von Datensätzen (neue Produkte, erweiterte oder geänderte Produkte, …)
- Angebotsausgabe in Excel-Format – durch die weite Verbreitung von Excel ergibt sich fast auf jedem Computer die Möglichkeit das Angebot einzusehen und eventuell noch kleine Anpassungen vorzunehmen.
- LV-Eingabe: Leichtes Kopieren von eingegebenen Produkten, Importieren bereits vorhandener Bauteile, usw.

Alle diese Vorteile werden durch diese Programmierung voll ausgeschöpft.

Vorgehensweise zur Erstellung einer Verbindung zu einem Excel-Com-Server:
Über die Variant - Variable *Excel* wird zuerst die Verbindung zu der Excel-Anwendung hergestellt. Über *WorkBooks* wird ein Arbeitsblatt in Excel vorbereitet, in welches eine

[24] Borland C++ Online Hilfe

Vorlage aus einer Exceltabelle *Exp_Datenbank_orig.xls* geladen wird. In der Anwendung kommt dieser Algorithmus für den Export der Datenbank in eine Exceltabelle vor. Nach dem Export und dem Speichern als .xls-File wird über die Funktionen *Excel.OleFunction("Quit");* und *Excel = Unassigned;* die Verbindung beendet und getrennt. Erkennen kann man das an diesem Beispiel:

```
Table1->Active = false;
Variant Excel;
try
 {
  Excel = GetActiveOleObject("Excel.Application");
 }
catch(...)
 {
  Excel = CreateOleObject("Excel.Application");
 }
//  Excel.OlePropertySet("Visible", true);
try
 {
  Variant WorkBooks = Excel.OlePropertyGet("WorkBooks");
  //WorkBooks.OleFunction("Add");
  WorkBooks.OleFunction("Open",Dir + "\\files\\Exp_Datenbank_orig.xls");
 }
catch(...)
 {
 }
Variant ActiveWorkBook = Excel.OlePropertyGet("ActiveWorkbook");
Variant WorkSheets = Excel.OlePropertyGet("Worksheets");
Variant WorkSheet = WorkSheets.OlePropertyGet("Item", 1);
WorkSheet.OleFunction("Activate");
Table1->Open();
Query1->Close();
Query1->SQL->Clear();
AnsiString ASSQLCommand = "SELECT * FROM Angebot_DB_Aktuell ORDER BY ID";
Query1->SQL->Add(ASSQLCommand);
Query1->Open();
Query1->Active = true;
Query1->First();
//MaxWert für Fortschrittsanzeige setzen
ProgressBar1->Max = Query1->RecordCount;
int iRow, iCol;
try
 {
  //Beschreiben der Excelliste
  for (iRow=2; iRow <= AnzahlZeilenExpImp; iRow++)
   {
     //Fortschrittsanzeige +1 setzen
```

```
    ProgressBar1->StepIt();
    for (iCol=1; iCol <=26; iCol++)
      {
       Variant Range = WorkSheet.OlePropertyGet("Cells", iRow, iCol);
       Range.OlePropertySet("Value", Query1->Fields->Fields[iCol-1]->AsString);
      }
    Query1->Next();
    //Bei Eof Abbruch des Exports
    if (Query1->Eof) break;
  }
 //ShowMessage("Daten wurden in Excel exportiert!");
 }
catch(...)
 {
  ShowMessage("Fehler bei Datenexport! in Zeile: "+IntToStr(iRow)+" Spalte: "+
IntToStr(iCol));
 }
try
 {
 ActiveWorkBook.OleFunction("SaveAs", Dir + "\\Exp_Datenbank.xls");
 }
catch(...)
 {
 }
Excel.OleFunction("Quit");
Excel = Unassigned;
Table1->Active = true;
//Fortschrittsanzeige auf 0 setzen
ProgressBar1->Position = 0;
}
```

4.5.7 Anzeigemöglichkeit des fertigen Angebotes

Mit Hilfe des kostenlosen Programms „FreePDF XP“ ist es möglich, die Tabelle aus dem Excel in ein PDF-File umzuwandeln und mit dem Adobe Reader anzuzeigen.

4.6 Aufgetretener Fehler beim Programmieren

Ein Problem entstand beim Hinzufügen eines DB Grid. Dies ist ein Objekt um Datenbankeinträge ohne viel Aufwand in einer Tabelle darstellen zu können. Beim Kompilieren wurde dann die *Fehlermeldung E2015 Mehrdeutigkeit zwischen 'Menus' und 'Excel_2k::Menus*' angezeigt.

Das Problem wurde mit Hilfe eines „Forums" im Internet gelöst. Dort hatte ein Benutzer ein ähnliches Problem. Reiht man die Includefiles so, dass "#include "Excel_2K_SRVR.h"" am Ende steht, so gibt der Compiler keinen Fehler mehr aus und die richtige geplante Funktion ist gegeben.

```
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Db.hpp>
#include <DBTables.hpp>
#include <Grids.hpp>
#include <OleServer.hpp>
#include <DBGrids.hpp>
#include "Excel_2K_SRVR.h"
```

4.7 InnoSetup

Inno-Setup bietet die Möglichkeit aus dem fertigen Programm ein Installationsfile mit allen dazugehörigen Dateien und Komponenten zu erstellen. Inno-Setup ist ein Open-Source-Installationsprogramm von jrsoftware.org - Jordan Russell's Software[25].

[25] Russell, Jordan: URL: www.jrsoftware.org, verfügbar am 10.02.2009

Dieses Programm ist skript-basierend, d.h. alle notwendigen Informationen müssen in einzelnen Kommandozeilen untereinander angeführt werden. Ähnlich wie bei INI-Files ist das Skript in mehrere Sektionen unterteilt, um eine bessere Übersichtlichkeit zu erhalten.

Die wichtigsten Bereiche sind:

- Setup: Hier werden grundsätzliche Einstellungen wie Programmname, Version, Verzeichnis, Setupname, usw. festgelegt.
- Files: Alle Dateien des Programms werden hier angegeben und der Installationsort wird hier festgelegt.
- Run: Abschließend kann man nun noch angeben, welche Programme am Ende der Installation gestartet werden sollen. Hier kann die Startreihenfolge definiert werden, Parameter können übergeben werden, usw.

Um ein einfaches Ändern eines fertigen Skriptfiles zu ermöglichen, kann der Benutzer Definitionen festlegen. Dabei können diese Variablen mit der richtigen Syntax im kompletten Skriptfile verwendet werden. Dies eignet sich besonders gut für Verzeichnisse, Namen und Versionen.

5 Die Anwendung

5.1 Aufbau einer Klimatechnikanlage

Die Gebäudeautomation wird in drei Ebenen unterteilt, die Feldebene, die Automationsebene und die Managementebene.

Die GLT (Gebäudeleittechnik) befindet sich auf der obersten Ebene, der Managementebene. Sie besteht aus einem Computer mit einer aufgesetzten Software, welche das Gebäude überwacht und steuert. Über eine Visualisierung werden alle technischen Vorgänge innerhalb dieses Gebäudes angezeigt. Dazu gehören Daten von Betriebszuständen diverser Anlagenteile wie:[26]

- Motore
- Lüftungsklappen
- Ventile
- Störmeldungen
- Schalterstellungen

aber auch direkte Messwerte wie:

- Temperatur
- Druck
- relative oder absolute Feuchte
- externe Sollwerte
- Verbrauchszählerstände

Die DDCs sind die eigentlichen Steuerungen des Gebäudes, welche direkt die Steuerungs- und Regelungsaufgaben im Bereich der Heizungs-, Lüftungs- und Lichtsteuerungen übernehmen. Sie entsprechen größtenteils einer speicherprogrammierbaren Steuerung (SPS) und besitzen eine feste interne Verdrahtung. Der Ausgangspunkt dieses Steuerungstypes ist aber die Gebäudeautomatisierung, nicht wie bei der SPS die Industrieautomatisierung. Der

26 Christof Hübner: Herrmann Merz: Thomas Hansemann; Gebäudeautomation: Kommunikationssysteme mit EIB/KNX, LON und BACnet, München, 2007

Schwerpunkt einer DDC liegt an ihrer ausgereiften Regelung. Über ein Programm wird der Ablauf der Anlage festgelegt.

Viele moderne DDCs sind Microcontrollersysteme mit einer entsprechenden Firmware. DDCs werden meist mit speziellen Programmiersprachen, die oft grafisch sind, programmiert, wobei die Syntax und der Umfang herstellerspezifisch sind.

Der prinzipielle Aufbau und wie die Kommunikation zwischen der Feldebene, DDC und der GLT funktioniert, kann anhand nachfolgender Grafik veranschaulicht werden:

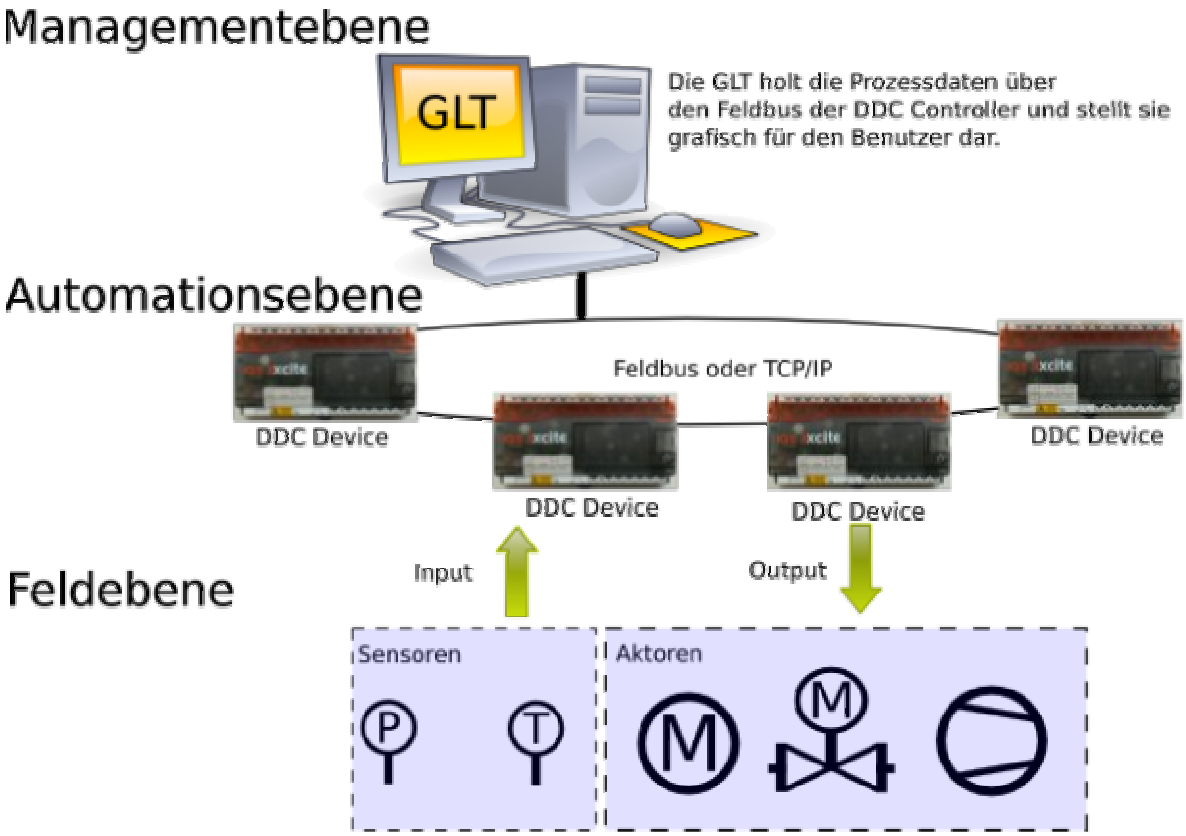


Abb. 4: Aufbau Feldebene - DDC - GLT[27]

[27]URL: http://homepage.swissonline.ch/inux-GmbH/, verfügbar am 09.04.2009

5.2 Allgemeiner Aufbau der Applikation

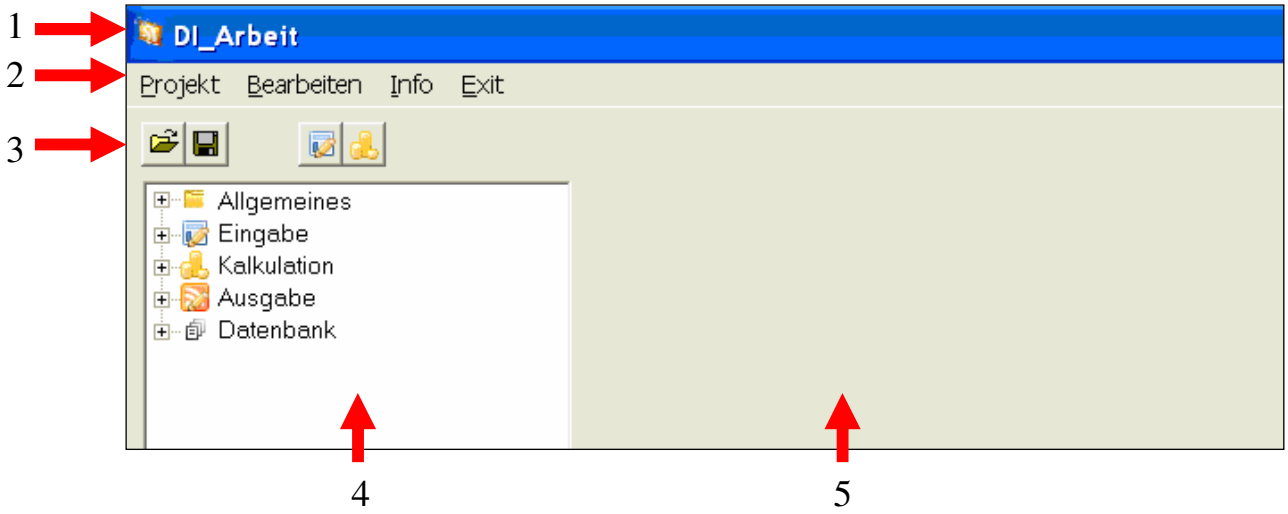


Abb. 5: Allgemeiner Aufbau der Applikation

1. Programmrahmen

   Enthält alle typischen Windows Eigenschaften, wie Minimieren, Maximieren und Beenden. Zuerst wird das Programmsymbol und anschließend der Programmname angezeigt. Hat man ein aktives Projekt gespeichert oder ein Projekt geöffnet, dann wird auch in dem Programmrahmen der Pfad mit dem aktuellen Projektnamen angezeigt. Wird eine Änderung im Projekt vorgenommen, wird das über einen Stern am Ende des Projektnamens signalisiert. Nach erfolgreicher Speicherung des Projektes wird der Stern automatisch entfernt. Wird das Programm minimiert, wird das entsprechende Programm inklusive dem zugehörigen Icon in der Windows - Taskleiste angezeigt.
2. Menüleiste

   Dient dem Benutzer, um allgemeine Einstellungen (wie z.B. Sprache) vorzunehmen oder allgemeine Information über das Programm zu erlangen.
3. Schnellstartleiste

   Wichtige und oft verwendete Stationen der Angebotserstellung können über das entsprechende Symbol in der Leiste sofort aufgeschaltet werden.
4. Projektübersichtsfenster

   Über dieses Fenster navigiert der Benutzer zu den einzelnen Stationen der

Angebotserstellung. Dieses Fenster ist immer sichtbar und es ermöglicht einen leichten Wechsel zwischen den Arbeitsfenstern.

5. Arbeitsbereich

   Hier werden je nach Auswahl aus dem Projektübersichtsfenster die entsprechenden Eingabe- oder Anzeigemasken aufgeschaltet.

Ein Wechseln zwischen den Stationen ist immer ohne Probleme machbar. Es gibt keinen Datenverlust durch ein Zurück- oder Vorwärtsnavigieren.

5.3 Die Taskleiste oder Infoleiste

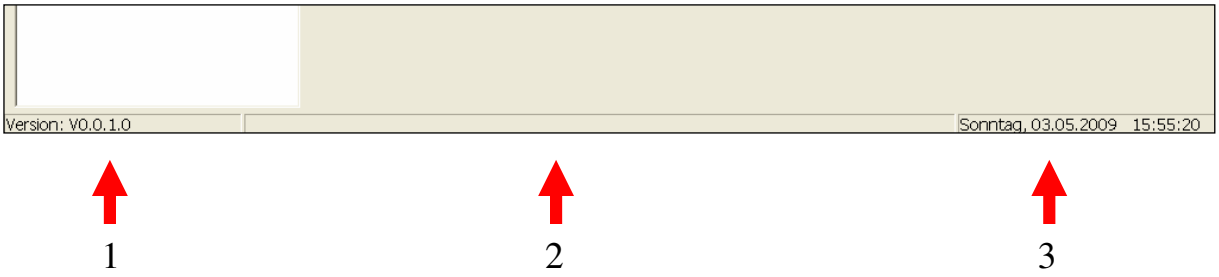


Abb. 6: Taskleiste

Diese Leiste ist in 3 Bereiche gegliedert:

1. Teilbereich: aktuelle Softwareversion
2. Teilbereich: aktuelles aktives Projekt
3. Teilbereich: aktuelles Datum und Uhrzeit

Der zweite Teilbereich wird auch als Fortschrittsanzeige bei diversen Funktionen genutzt. Wird ein Projekt geöffnet oder gespeichert, wird die Kalkulation berechnet oder auch das fertige Projekt erstellt, so erhält der Benutzer hier den aktuellen Fortschritt angezeigt.

5.4 Adressverwaltung

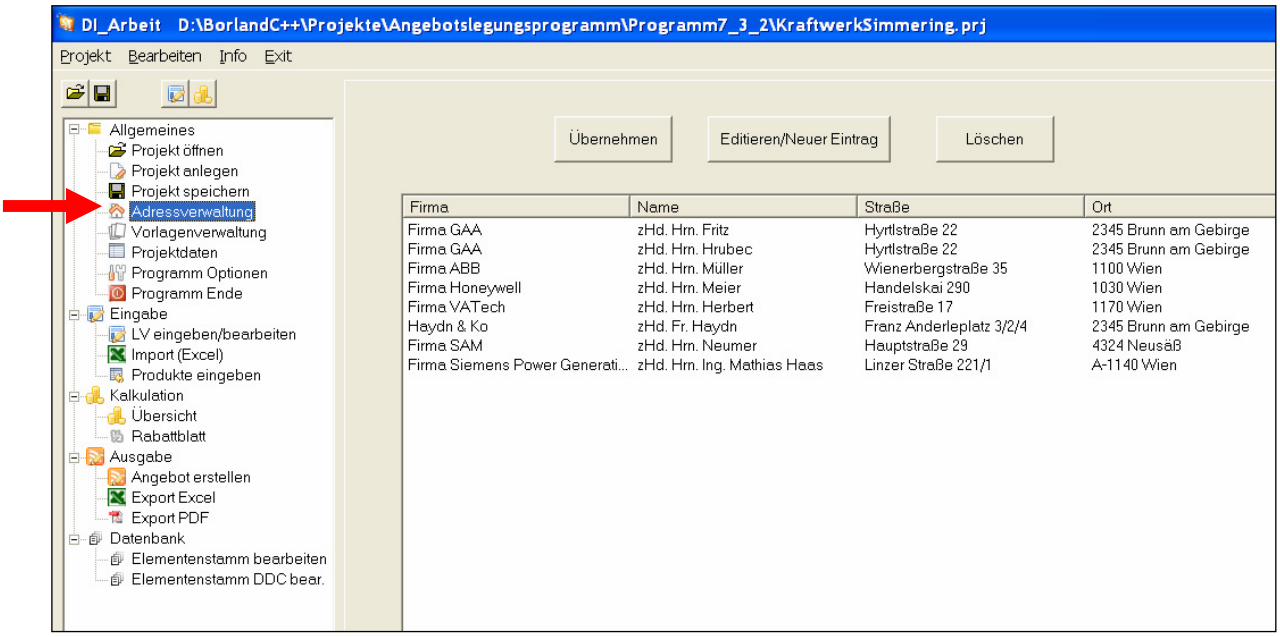


Abb. 7: Adressverwaltung

Alle Adressen können unter dem Punkt „Adressverwaltung“ gespeichert werden. Hier erhält der Benutzer die Möglichkeit, Änderungen an bestehenden Adressen vorzunehmen oder neue Firmenkontakte einzufügen. Über die Funktion „Übernehmen“ wird der selektierte Kontakt in das aktive Projekt übernommen. Über ein Bestätigungsfenster wird die erfolgreiche Aktion quittiert.

Abb. 8: Bestätigungsfenster Adresse übernommen

5.5 Projekt anlegen

Hier wird nun die übernommene Adresse aus der Adressverwaltung unter Anschrift automatisch eingefügt. Der Benutzer gibt nur noch die allgemeinen Projektdaten ein.

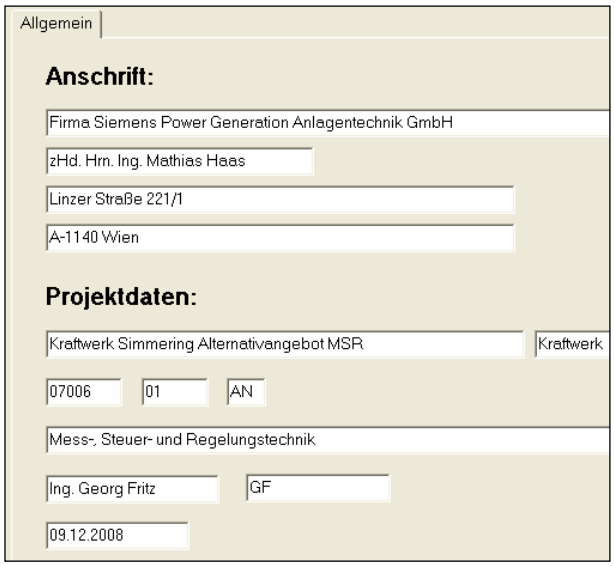


Abb. 9: Projekt anlegen

Jede Anlage kann auf Wunsch in mehrere Zonen (hier wird von Bauteilen gesprochen) eingeteilt werden. Jedem Bauteil wird ein separater Name zugewiesen und über Checkboxen wird festgelegt, wie viele Bauteile das Projekt besitzt. In diesem Beispiel besteht die Anlage aus drei Bauteilen: Turbinen Halle, Kesselhaus und Gasreduzierstation.

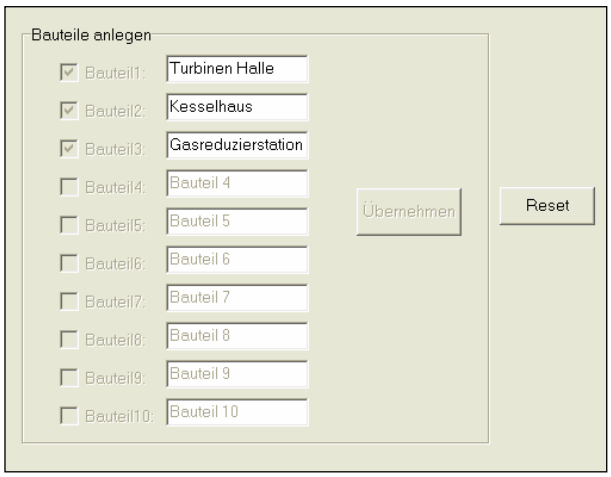


Abb. 10: Bauteile anlegen

5.6 Projekt öffnen/speichern

Über den Punkt „Projekt speichern“ im Projektübersichtsfenster oder über die Schnellstartleiste kann das aktuelle Projekt gespeichert werden. Wurde das aktuelle Angebot schon einmal gespeichert, dann steht im Programmrahmen und im zweiten Teilbereich der Statusleiste das Verzeichnis mit der abgespeicherten Datei. Über die Schnellstartleiste kann das Angebot unter diesem Namen abgespeichert werden. Soll jedoch unter einer anderen Datei das Projekt abgespeichert werden, dann muss über den Punkt „Projekt speichern“ im Projektübersichtsfenster diese Funktion ausgeführt werden. Dadurch erscheint ein Zusatzfenster, wo der Speicherpfad und der Name beliebig vergeben werden kann.

Alle Daten aus dem aktuellen Projekt werden in Dateiform auf dem Computer abgelegt. Standardmäßig ist das aktuelle Programmverzeichnis vorgewählt, der Dateiname kann beliebig gewählt werden. Die Dateinamenserweiterung (engl. „filename extension“) oder auch oft als Dateiendung bezeichnet, lautet „prj“. Mögliche Projektdateien ergeben sich daraus: KraftwerkSimmering.prj, HeizwerkMadeira.prj, FreizeitanlageBukarest.prj usw.
Durch diese Programmfunktion ist es leicht, ein bereits begonnenes Projekt auf einem beliebig anderen Rechner fortzusetzen oder ein altes Projekt als Vergleich wieder heranzuziehen. Zusätzlich werden die aktuellen Daten gesichert um keinen Datenverlust durch einen Programm- oder Rechnerabsturz zu erleiden.
Gespeichert werden aber nur statische Daten, das heißt die Ergebnisse der Kalkulation, werden in diesem File nicht mitgespeichert. Dazu ist es notwendig, das fertige Angebot in einer extra Datei abzuspeichern. Dies wird aber im Regelfall immer automatisch gemacht.

Über einen Stern im Programmrahmen wird angezeigt, dass in dem Projekt eine Änderung erfolgt ist. Dies soll als zusätzliche optische Hilfe dienen und der Benutzer weiß somit, dass er vor Beendigung des Programms oder bevor er ein altes Projekt laden möchte, sein aktives Projekt speichern muss bzw. sollte, um keinen Datenverlust zu erleiden. Wird das Angebotslegungsprogramm beendet, überprüft das Programm den Speicherstatus und schaltet gegebenenfalls das entsprechende Fenster für den Benutzer auf. Dadurch kann ein Speichern seines aktiven Projektes gar nicht vergessen werden.

Über den Punkt „Projekt öffnen“ kann ein zuvor gespeichertes Projekt wieder geöffnet werden. Alle Daten werden aus der Datei geladen und dem Benutzer in den unterschiedlichen Projektierungsfenstern zur Verfügung gestellt. Ohne weiteren Aufwand kann an dem Angebot weitergearbeitet werden. Der Benutzer hat jedoch auch die Möglichkeit, sich ein altes Projekt mit eventuell erneuerten Preisen neu kalkulieren zu lassen oder ein altes Projekt so umzuarbeiten, dass ein neues Angebot daraus erstellt werden kann. So erspart sich der Anbotsersteller Zeit, da er Teile oder vielleicht so gar den Grossteil des bereits bestehenden Anbotes übernehmen kann.

Eine Fortschrittanzeige im zweiten Teilbereich in der Statusleiste zeigt dem Benutzer, wie weit die Funktion Speichern/Laden (gewünschte Operation) fortgeschritten ist.

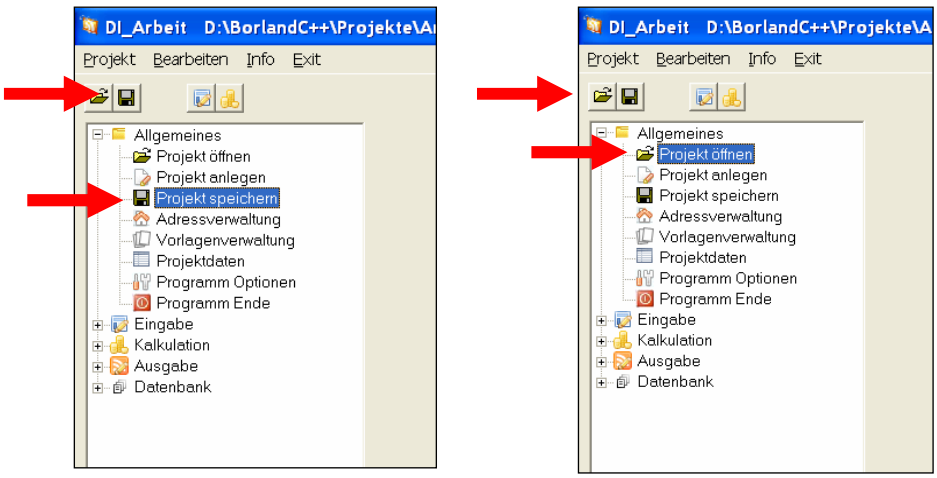


Abb. 11: Projekt speichern/öffnen

Nach dem Öffnen eines Projektes wird abgefragt, ob das LV (Leistungsverzeichnis) geladen werden soll, oder ob nur die allgemeinen Projektdaten, wie Kundenanschrift, Projektname und -nummer, Name der Bauteile, Rabatblatt, usw. übernommen werden sollen.
Soll aber z.B. ein komplettes altes Projekt geladen und mit eventuell neuen Preisen kalkuliert werden, dann muss diese Abfrage mit „Ja“ bestätigt werden.

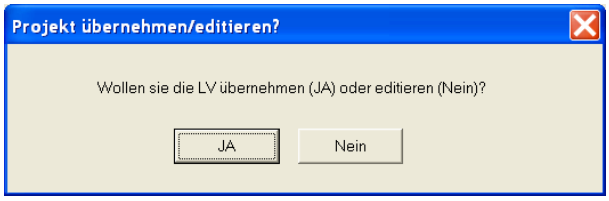


Abb. 12: Bestätigungsfenster LV übernehmen?

5.7 LV eingeben/bearbeiten

Unter dem Punkt „LV eingeben/bearbeiten“ (Leistungsverzeichnis) werden alle Elemente der einzelnen Bauteile eingegeben.

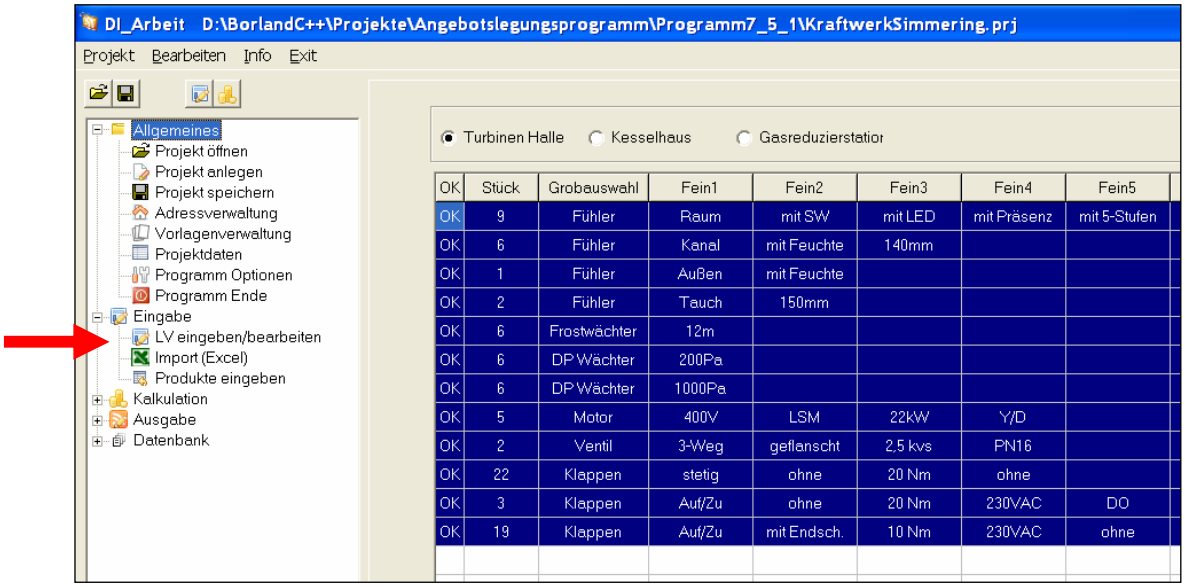


Abb. 13: Leistungsverzeichnis

Vorgehensweise:

- Bauteil auswählen
- gewünschte Stückanzahl eintragen
- über die Grobauswahl den Überbegriff wählen
- über mehrere Feinauswahlfelder wird das entsprechende Element immer genauer definiert.

Wird das Grobauswahlfeld mit dem Mauszeiger angewählt öffnet sich ein Pull-down-Menü mit den entsprechenden Überbegriffen. Diese sind alle in der Datenbank hinterlegt und werden entsprechend herausgefiltert. Hat man den Überbegriff gewählt, so wird nun über das Feinauswahlfeld 1 sein gewünschtes Feld genauer definiert. Auch hier öffnet sich ein Pull-down-Menü, wo es entsprechende Vorgaben zu dem zuvor ausgewählten Überbegriff gibt. Man verfährt nach diesem System bis das gewünschte Element bis ins Detail genau definiert worden ist. Das erkennt man daran, dass sich die Hintergrundfarbe ändert, in der ersten Spalte wird ein „OK“ angezeigt und es werden der Gerätetyp und der Hersteller angezeigt. Alle diese

Informationen werden direkt aus der Datenbank eingetragen. Diese ganze Prozedur ist dynamisch, d.h. die Auswahl aus den Pull-down-Menüs wird während der Laufzeit direkt aus der Datenbank abgefragt. Wird nun in der Datenbank ein neues Element eingetragen, so kann man es sofort im Leistungsverzeichnis verwenden.

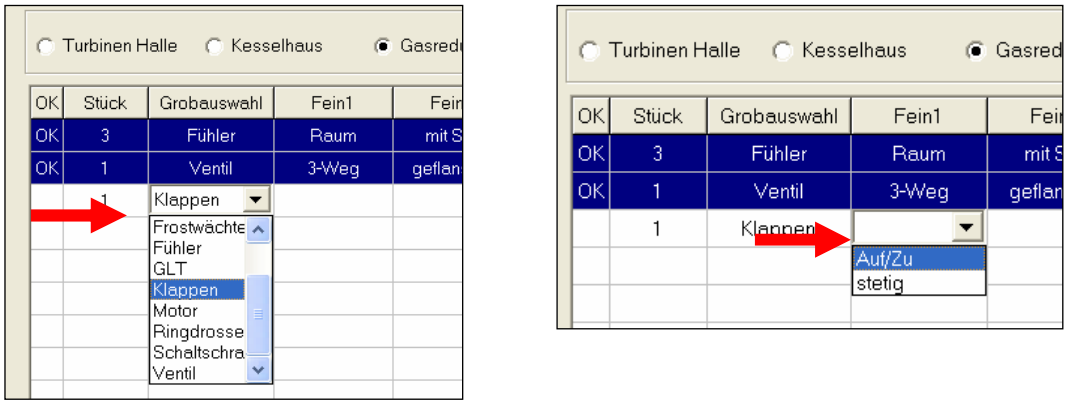


Abb. 14: Auswahlmöglichkeit Leistungsverzeichnis

Wurde ein Element genau definiert sieht man nun den Gerätetyp, den Hersteller und noch weitere Informationen. Diese sind dann für die weiteren Kalkulationsarbeiten notwendig.
In diesem Beispiel wären das folgende Informationen:
Überbegriff: Klappen
Gerätetyp: N20230
Hersteller: Honeywell
Anzahl der digitalen Ausgänge von CPU benötigt: 1
Rabattgruppe Einkauf: A
Rabattgruppe Verkauf: A
Listenpreis Type1: 104,94€
Betriebsmittelgruppe: FuG (Fühler&Geber)
In der letzen Spalte wird noch für weitere Informationen der Ausschreibungstext angezeigt, welcher aber normalerweise erst im fertigen Angebot interessant ist.

Eine weitere Funktion bietet der Export und Import des LVs in Excel. Dies dient als zusätzliche Funktion um einerseits die Vorteile von Excel ausnützen zu können, andererseits aber auch um schon bestehende LVs einbinden zu können. Oft ist es ja so, dass es bei jedem Klimaanlagenprojekt ein Nebengebäude gibt, welcher ebenfalls klimatisiert sein soll. Gerade solche Standardkleinanlagen in großen Projekten können so ohne viel Aufwand in ein

Angebot integriert werden. Aber auch andere, eventuell größere Anlagenteile, welche ähnlich einem Anlageprojekt sind, können einfach in das neue Projekt eingebunden werden. Standardmäßig wird ein Export in das aktuelle Programmverzeichnis unter den Namen „Exp_LV.xls“ gespeichert. Auch ein Import wird aus der Datei „Exp_LV.xls“ durchgeführt. Im zweiten Teilbereich der Taskleiste wird der aktuelle Fortschritt des Exports/Imports angezeigt.

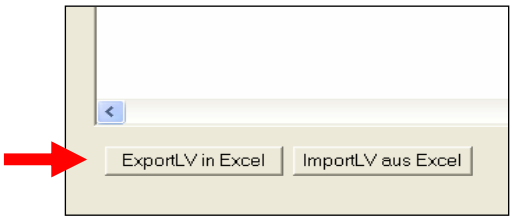


Abb. 15: Export-/Importfunktion

5.8 Datenbank - Elementenstamm bearbeiten

In der Datenbank sind alle Motore, Ventile, Fühler, usw. aufgelistet. Alle Informationen zu den einzelnen Geräten, wie Gerätetyp, Listenpreis, Angebotstext, Anzahl der digitalen/analogen Ein- bzw. Ausgänge und viele weitere notwendige Daten sind dort gespeichert. Dies wird in Tabellenform angezeigt. Über den Navigatorbalken kann man zwischen den einzelnen Datensätzen navigieren, neue Datensätze anlegen oder auch Daten ändern oder löschen.

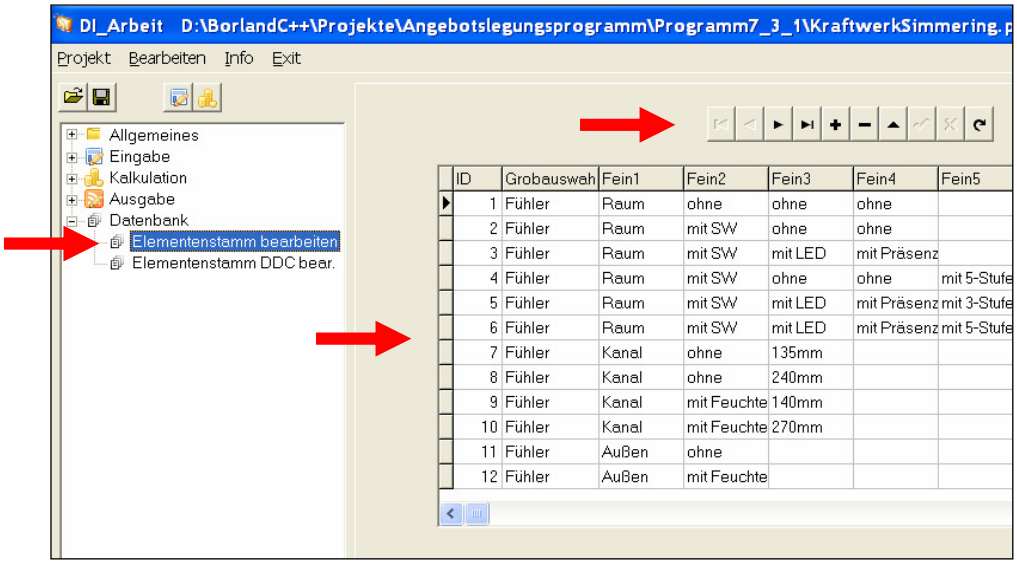


Abb. 16: Datenbank

Wird der nächste Datensatz oben in der Tabellenform ausgewählt, erhält man auf einen Blick im unteren Bildschirmbereich die Anzeige von allen wichtigen Daten. Auch hier ist es möglich den Datensatz ohne Probleme zu editieren.

Über den Button „In Excel exportieren“ und „DatenAusExcel importieren“ kann die komplette Datenbank in das Programm Microsoft Excel exportiert und importiert werden.

Die Verbindung vom C++Builder zu MS-Officeanwendungen erfolgt über sogenannte COM-Server. Es gibt eine Vielzahl an Servern, wobei hier nur die Excel-Com-Server Verbindung verwendet wird. Über diese Verbindung werden Daten ausgetauscht, wodurch folgende Vorteile für den Benutzer entstehen:

- Export der Datenbanken in MS-Excel: Durch einen kompletten Export der Datenbank in Excel wird dem Benutzer die Möglichkeit gegeben alle Vorteile des Excels zu nutzen, um darin Daten zu editieren, zu löschen oder Datensätze hinzuzufügen.
- Import der Datenbanken aus Excel: Dadurch hat der Benutzer die Möglichkeit, seine angepassten Daten ohne großen Aufwand wieder in die Programmdatenbank zu importieren.

Der Fortschritt des Imports/Exports wird über einen Balkenanzeige dem Benutzer angezeigt.

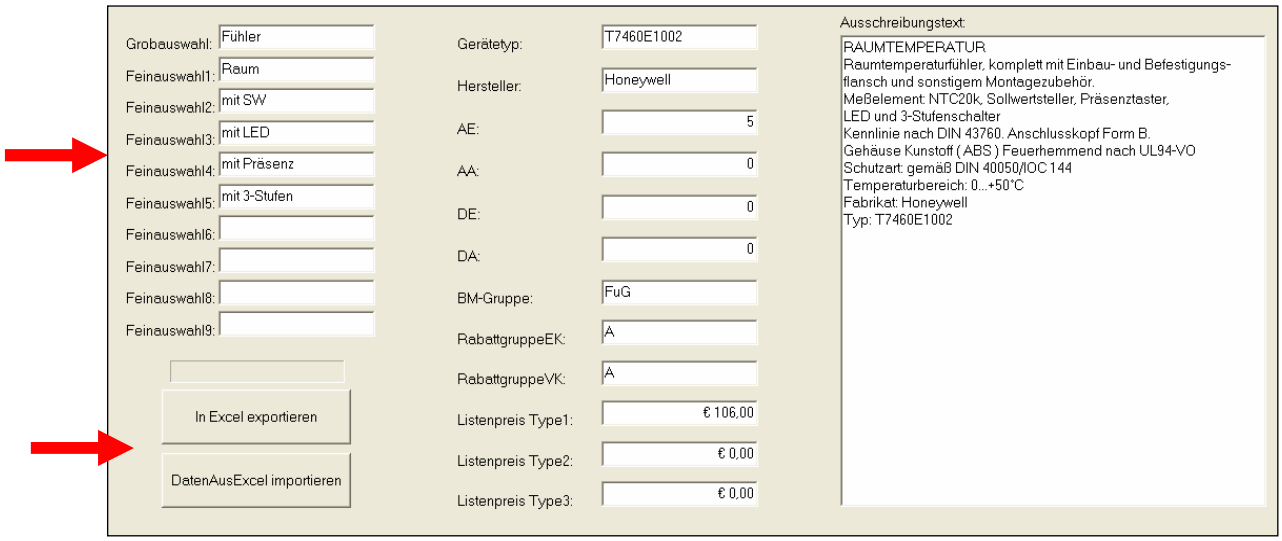


Abb. 17: Datenbank Detailansicht

5.9 Datenbank - Elementenstamm DDC bearbeiten

Die zweite Datenbank beinhaltet alle DDC (Direct Digital Control) - Geräte. Das sind Computer ähnliche elektronische Baugruppen, welche für Steuerungs- und Regelungsaufgaben in der Gebäudeautomatisierung eingesetzt werden. Die Funktionalität dieser Datenbank ist identisch mit der zuvor beschriebenen Datenbank, sowohl die Bearbeitung als auch die Export/Importfunktion in Excel.

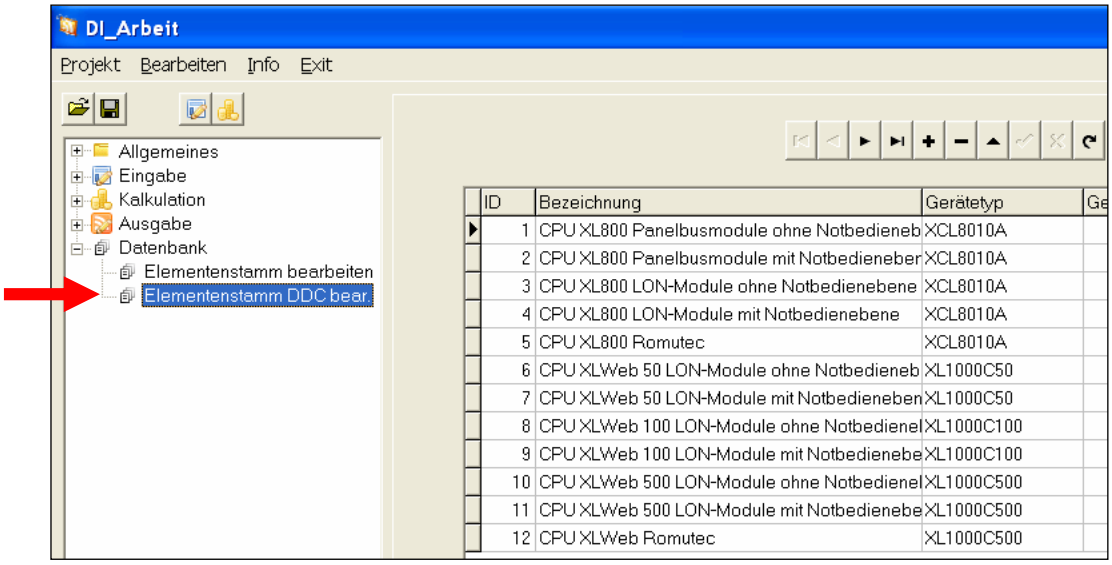


Abb. 18: Datenbank DDC

5.10 Rabattblatt

Überall geht es um Kosten. Sie sind ein wichtiger Bestandteil eines jeden Angebots und oft, wenn nicht sogar immer, bestimmt der Preis, ob das Angebot überhaupt näher betrachtet wird oder nicht. Unternehmen erhalten bei gewissen Herstellern oft Rabatte auf den Einkaufspreis. Dadurch kann das Unternehmen teilweise diese Prozente dem Kunden über das Angebot weitergeben. Dies kann auch der entscheidende Ausschlag für den Erhalt des Auftrages sein. Der Angebotsleger hat die Möglichkeit, über das Rabattblatt, auf einzelne Produktgruppen einen Nachlass in Prozenten zu vergeben. In diesem Rabattblatt werden einerseits die Rabatte auf den Einkaufspreis, aber auch die Rabatte oder Aufschläge auf den Verkaufspreis

eingetragen. So kann das Unternehmen beim Erstellen des Angebots selbst entscheiden, wie viel es von ihren Prozenten an den Kunden weitergeben möchte. Diese doppelte Eingabe wird später benötigt, um den Preisunterschied zwischen dem Gesamteinkaufspreis und dem Gesamtverkaufspreis zu erhalten.

Da der Einkaufsrabatt, welchen das Unternehmen bei seinen Lieferanten bekommt, für alle Projekte gilt, sind auch die Daten unter einer eigenen Datei abgelegt. Standardmäßig ist das unter dem Programmverzeichnis mit dem Unterordner „files“ die Datei „Rabatt.blatt“. Wird also hier eine Änderung vorgenommen, wird dies in die Datei geschrieben.

Der Verkaufsrabatt kann aber von Projekt zu Projekt unterschiedlich sein. Aus diesem Grund werden diese Daten auch unter dem aktuellen Projekt abgespeichert. Wird ein neues Projekt erstellt, werden standardmäßig die Einkaufsrabatte übernommen.

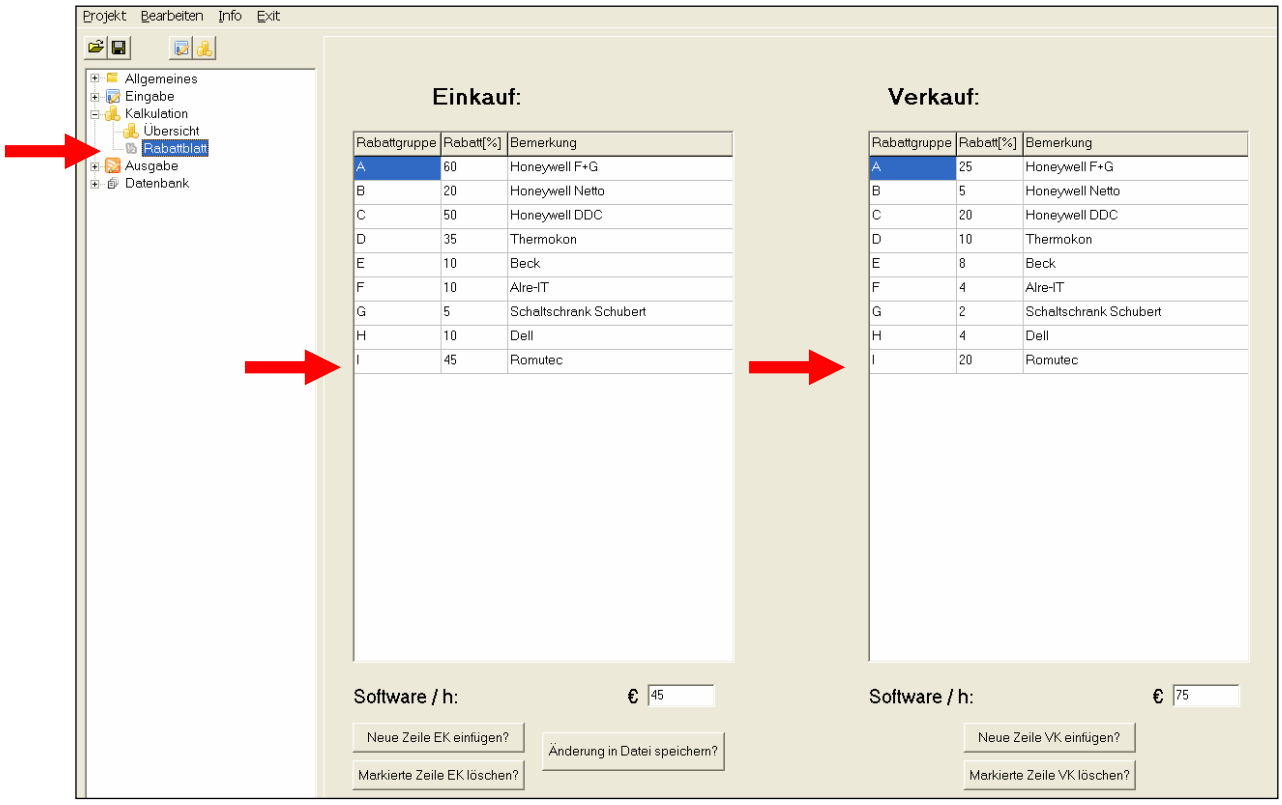


Abb. 19: Rabattblatt

5.11 Kalkulation – Übersicht

In diesem Fenster werden bauteilmäßig die einzelnen Elemente des LVs zusammengefasst. Über den Button „Datenübernahme“ werden alle Daten aus dem LV in die Kalkulationsübersicht eingefügt und entsprechend ihrer Betriebsmittelgruppe sortiert. Nun sind alle notwendigen Daten ersichtlich, welche für die Kalkulation benötigt werden.

Folgende Betriebsmittelgruppen stehen pro Bauteil zur Auswahl:

- F&G: Fühler und Geber
- DDC: Steuerungen
- SS: Schaltschrank
- SSBG: Beistellgeräte Schaltschrank

Die Registerlaschen GLT (Gebäudeleittechnik) und SW+IBS (Software und Inbetriebsetzung) sind pro Angebot nur einmal vorgesehen, da sie sich global auf die gesamte Anlage beziehen.

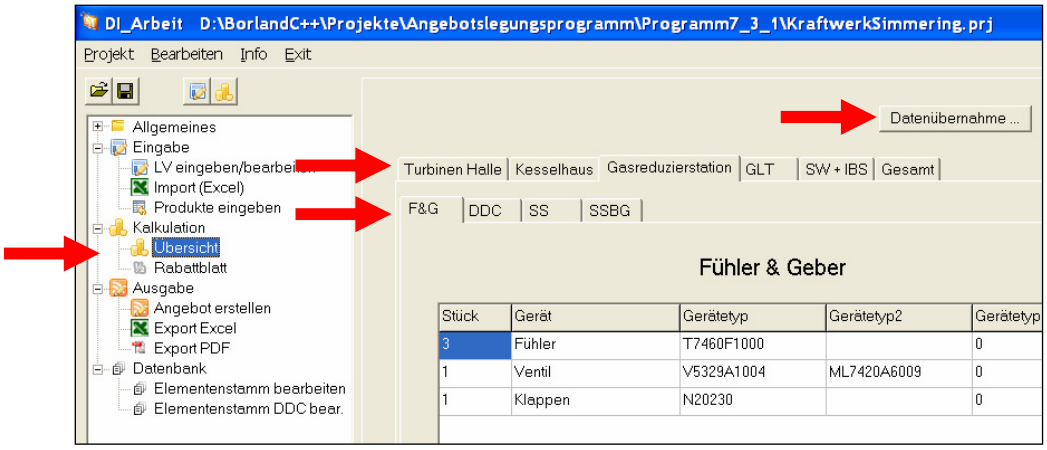


Abb. 20: Kalkulation - Übersicht

Nun besteht für den Benutzer die Möglichkeit seine Eingaben nochmals zu überprüfen. Weiters muss nun bei den Steuerungen (DDC) der Steuerungstyp gewählt werden.

Die Auswahlmöglichkeit erfolgt aus den Daten der zweiten Datenbank. Durch diese entstehende Dynamik ist es jederzeit möglich, einen neuen Steuerungstyp in die Datenbank einzutragen, welcher dann sofort in dem Projekt verfügbar ist. Die Auswahl der Steuerung

muss für jeden Bauteil erfolgen. Wird auf eine Auswahl vergessen, kann die Kalkulation nicht durchgeführt werden und eine entsprechende Meldung erscheint.

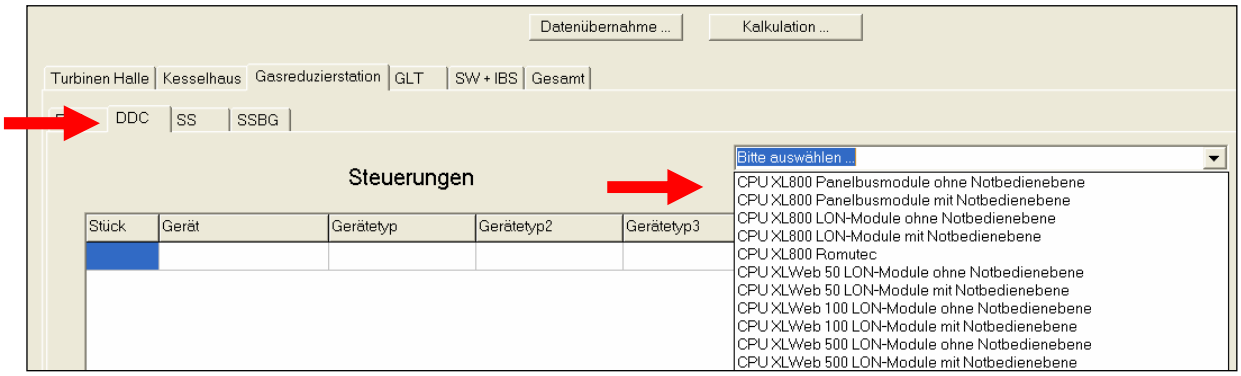


Abb. 21: CPU Auswahl

In der Registerlasche SW+IBS muss noch die Projektierung für die Software, Dokumentation und Inbetriebnahme durchgeführt werden. Der Aufwand dafür ergibt sich aus der Größe der Anlage. Als Rechengrundlage wird die Gesamtzahl der Datenpunkte herangezogen. Es besteht die Möglichkeit hier die eigenen tatsächlichen Kosten mit einem Zuschlag zu versehen, um auch hier einen Gewinn zu erzielen. Dies wird über die eigenen Eingabefelder ermöglicht. Am Ende werden noch die Reisekosten für Anfahrt- und Unterkunft eingetragen.

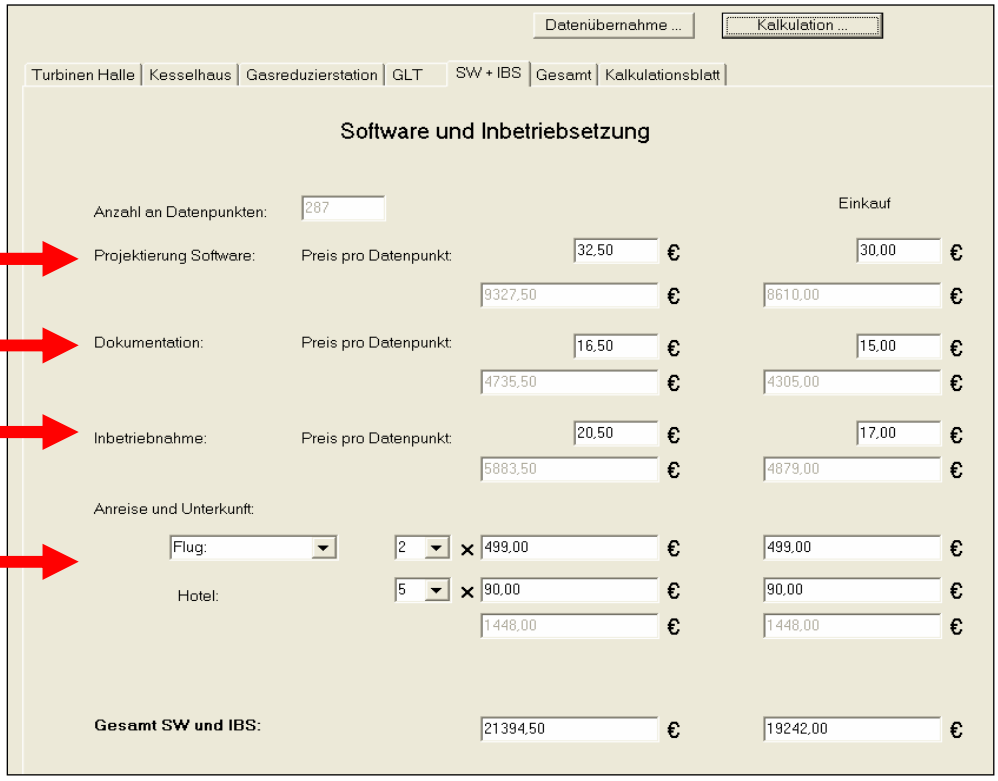


Abb. 22: Eingabe für Software, Dokumentation und Inbetriebnahme

Nachdem alle Angaben eingetragen wurden, wird über den Button „Kalkulation“ das Angebot gerechnet.

Zuerst legt das Programm fest, wie viele CPUs und wie viele zugehörige E/A-Module benötigt werden. Welcher CPU Typ Verwendung findet, wurde zuvor schon festgelegt. Hat der Benutzer auf diese Eingabe vergessen, so wird er bei der Kalkulation darauf hingewiesen und er kann dies nachholen. Um dies zu erreichen, wird die Anzahl der Datenpunkte ermittelt, welche danach auch für die Lizenz der „Grafischen Leitwarte“ benötigt werden. Über die Datenpunktanzahl erhält man Aufschluss, welche Module und wieviele in den jeweiligen Bauteilen verwendet werden. Dies ist natürlich von CPU zu CPU verschieden. Je nach Typ werden die entsprechenden Module und auch die Anzahl der Module vorgesehen.

In diesem Beispiel wurde für die Turbinenhalle die CPU XL800 Romutec ausgewählt. Die dazugehörigen E/A Module sind:

- AI LON XFL821A für die analogen Eingänge (AE)
- AO-Modul BAH4000 für die analogen Ausgänge (AA)
- DO/DI-Modul BDH1401 für die digitalen Eingänge (DE) und Ausgänge (DA)
- DI-Modul BLM1001 für die restlichen digitalen Eingänge (DE)

Die benötigten Ein- und Ausgänge werden addiert und die Anzahl der Module dementsprechend in das Angebot eingebunden. Die digitalen Ausgangsmodule enthalten aber auch digitale Eingänge, welche natürlich Verwendung finden. Deren Anzahl reduziert dann die Gesamtanzahl an digitalen Eingängen, welche für die Berechnung der digitalen Eingangsmodule herangezogen wird.

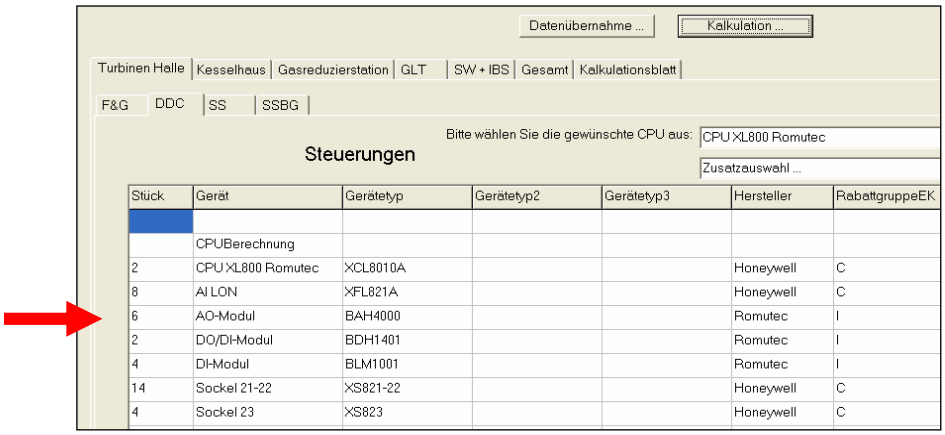


Abb. 23: Berechnung der CPU und der Module

Nach erfolgreicher Bestimmung der richtigen Module und der entsprechenden Anzahl wird nun der Preis kalkuliert.

Dabei werden alle Geräte (Elemente) samt Stückzahl und ihrem zugehörigen Ausschreibungstext aufgelistet. Auf Wunsch werden der Einzel- und Gesamtpreis, sowie am Ende der Liste zusammenfassend die Gruppenpreise angezeigt. Die Preise werden über die entsprechende Rabattgruppe berechnet. Wird der Schaltschrank an einen Subanbieter weitervergeben, so kann direkt der Einkaufspreis samt einem Aufschlag eingegeben werden. Daraufhin werden alle Schaltschrankdaten ausgeblendet und nicht mehr im Gesamtpreis berücksichtigt. Zusätzlich steht dem Bediener die Möglichkeit offen, bei Bedarf einen Rabatt auf das gesamte Projekt zu vergeben.

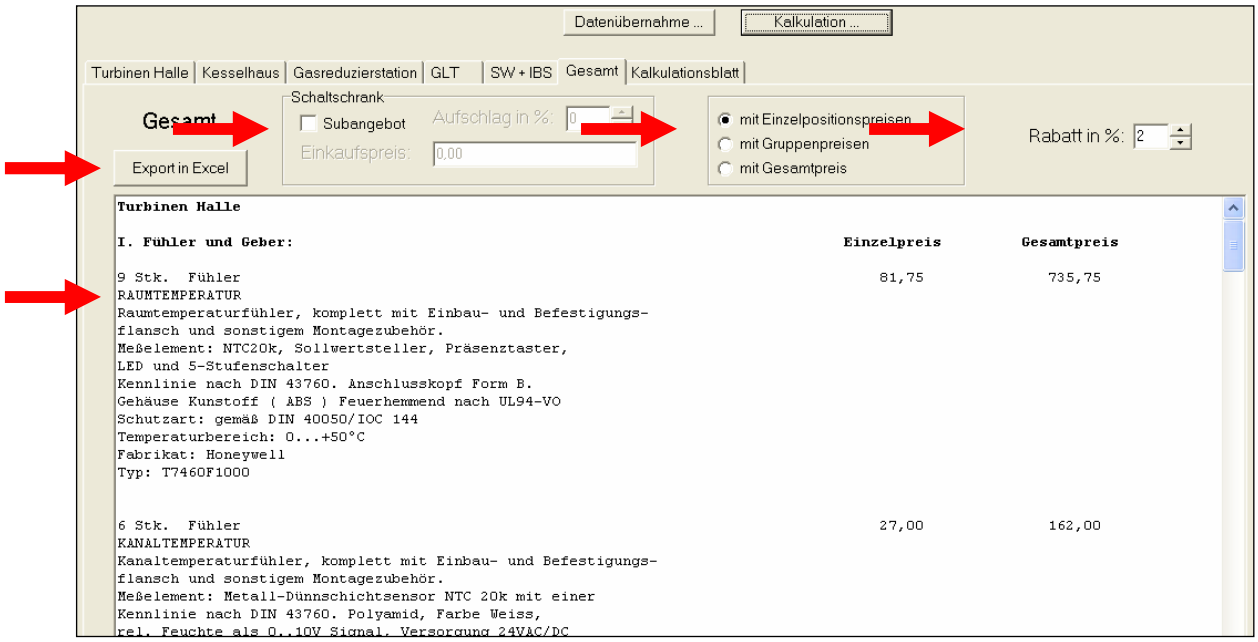
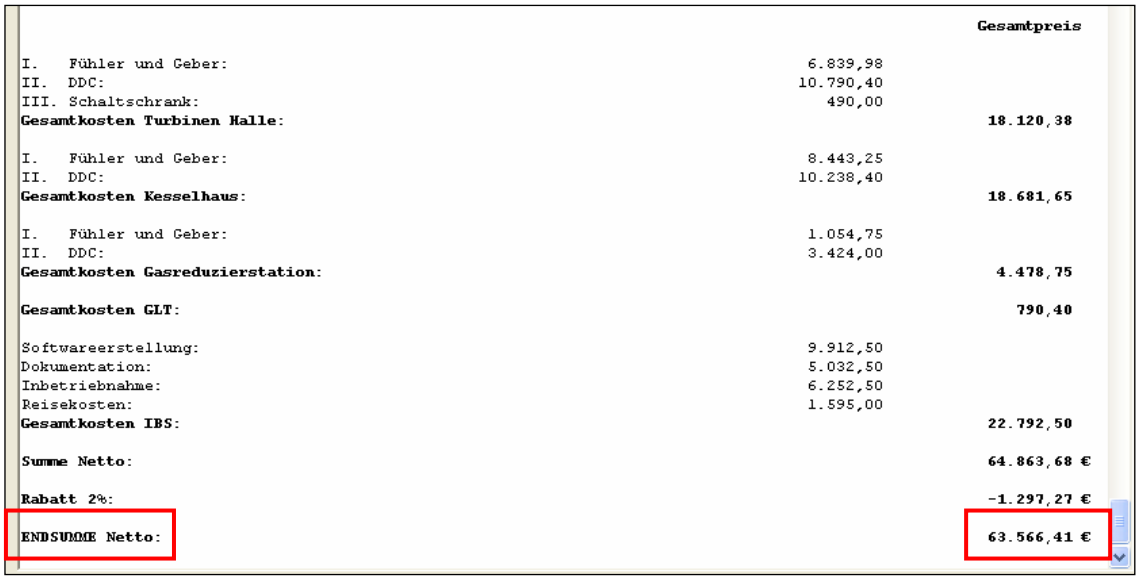


Abb. 24: Gesamtkalkulation

Am Ende des Dokuments werden auf Wunsch alle Gruppenpreise aufgelistet und die Summe gebildet. Zusätzlich zu den Gerätekosten werden noch die Inbetriebsetzungskosten, welche aus Softwareerstellung, Dokumentation, Inbetriebnahme und Reisekosten bestehen, hinzuaddiert. Die Endsumme wird abzüglich des Kundenrabatts angezeigt.

Ist der Bearbeiter mit dem Ergebnis zufrieden, so kann das komplette Angebot in ein Excelfile exportiert werden. Dort ist das komplett fertige Angebot, samt allgemein gültigen Texten, wie Zahlung, Gewährleistung, Lieferbedingungen usw. nun erhältlich. Bei Bedarf kann dieses

dann sofort an den Kunden per Email übermittelt werden. Der Umweg über Excel wurde gewählt, da nahezu jeder Kunde das Programm „Microsoft Excel“ zu Verfügung hat.



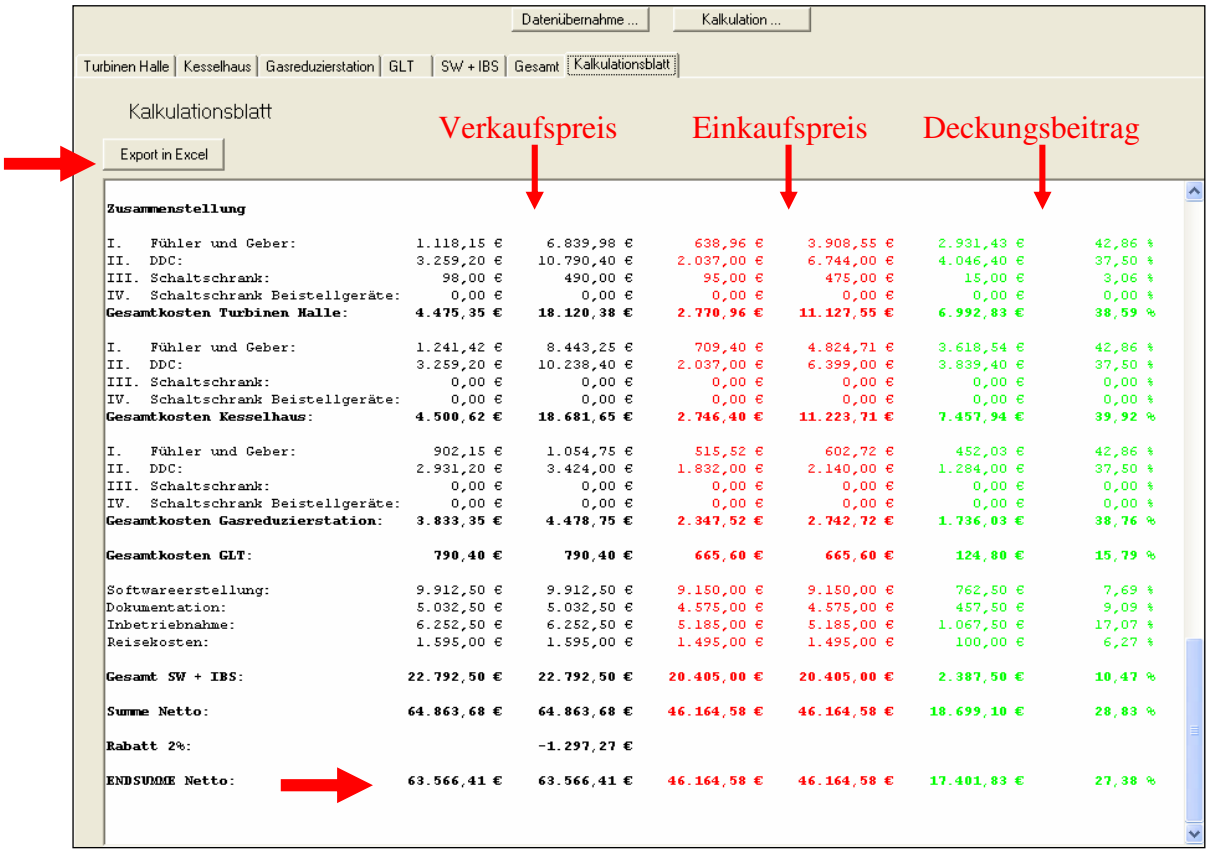
		Gesamtpreis
I. Fühler und Geber:	6.839,98	
II. DDC:	10.790,40	
III. Schaltschrank:	490,00	
Gesamtkosten Turbinen Halle:		**18.120,38**
I. Fühler und Geber:	8.443,25	
II. DDC:	10.238,40	
Gesamtkosten Kesselhaus:		**18.681,65**
I. Fühler und Geber:	1.054,75	
II. DDC:	3.424,00	
Gesamtkosten Gasreduzierstation:		**4.478,75**
Gesamtkosten GLT:		**790,40**
Softwareerstellung:	9.912,50	
Dokumentation:	5.032,50	
Inbetriebnahme:	6.252,50	
Reisekosten:	1.595,00	
Gesamtkosten IBS:		**22.792,50**
Summe Netto:		**64.863,68 €**
Rabatt 2%:		**-1.297,27 €**
ENDSUMME Netto:		**63.566,41 €**

Abb. 25: Zusammenfassung Gruppenpreise Gesamtkalkulation

Auf dem Kalkulationsblatt werden alle Geräte und Einzelpositionen in Kurzform aufgelistet. Der Hintergedanke bei dieser Übersicht besteht darin, dass hier alle Einzelpreise und Gesamtpreise von Verkauf, sowie Einkauf angezeigt werden. Darauf wird der entsprechende Deckungsbeitrag berechnet, wobei dieser in Euro und Prozenten angegeben wird. Dieser Beitrag dient zur Deckung aller Fixkosten für das Projekt. Der Auftragnehmer hat somit bei einer Auftragsverhandlung alle seine Kosten auf einen Blick und kann leicht auf Kundenwünsche reagieren.

Zusammenfassend werden am Ende dieses Blattes alle Gruppenpreise, sowie Inbetriebsetzungskosten aufgelistet. Daraus wird die Summe, abzüglich des Kundenrabattes, gebildet. Nun ist das komplette Anlagenprojekt erstellt bzw. kalkuliert. Gleichzeitig ist der Deckungsbeitrag erkennbar.

Auch hier besteht die Möglichkeit, sich das fertige Kalkulationsblatt in ein Excel File zu exportieren.



Zusammenstellung

I. Fühler und Geber:	1.118,15 €	6.839,98 €	638,96 €	3.908,55 €	2.931,43 €	42,86 %
II. DDC:	3.259,20 €	10.790,40 €	2.037,00 €	6.744,00 €	4.046,40 €	37,50 %
III. Schaltschrank:	98,00 €	490,00 €	95,00 €	475,00 €	15,00 €	3,06 %
IV. Schaltschrank Beistellgeräte:	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 %
Gesamtkosten Turbinen Halle:	**4.475,35 €**	**18.120,38 €**	**2.770,96 €**	**11.127,55 €**	**6.992,83 €**	**38,59 %**
I. Fühler und Geber:	1.241,42 €	8.443,25 €	709,40 €	4.824,71 €	3.618,54 €	42,86 %
II. DDC:	3.259,20 €	10.238,40 €	2.037,00 €	6.399,00 €	3.839,40 €	37,50 %
III. Schaltschrank:	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 %
IV. Schaltschrank Beistellgeräte:	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 %
Gesamtkosten Kesselhaus:	**4.500,62 €**	**18.681,65 €**	**2.746,40 €**	**11.223,71 €**	**7.457,94 €**	**39,92 %**
I. Fühler und Geber:	902,15 €	1.054,75 €	515,52 €	602,72 €	452,03 €	42,86 %
II. DDC:	2.931,20 €	3.424,00 €	1.832,00 €	2.140,00 €	1.284,00 €	37,50 %
III. Schaltschrank:	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 %
IV. Schaltschrank Beistellgeräte:	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 %
Gesamtkosten Gasreduzierstation:	**3.833,35 €**	**4.478,75 €**	**2.347,52 €**	**2.742,72 €**	**1.736,03 €**	**38,76 %**
Gesamtkosten GLT:	**790,40 €**	**790,40 €**	**665,60 €**	**665,60 €**	**124,80 €**	**15,79 %**
Softwareerstellung:	9.912,50 €	9.912,50 €	9.150,00 €	9.150,00 €	762,50 €	7,69 %
Dokumentation:	5.032,50 €	5.032,50 €	4.575,00 €	4.575,00 €	457,50 €	9,09 %
Inbetriebnahme:	6.252,50 €	6.252,50 €	5.185,00 €	5.185,00 €	1.067,50 €	17,07 %
Reisekosten:	1.595,00 €	1.595,00 €	1.495,00 €	1.495,00 €	100,00 €	6,27 %
Gesamt SW + IBS:	**22.792,50 €**	**22.792,50 €**	**20.405,00 €**	**20.405,00 €**	**2.387,50 €**	**10,47 %**
Summe Netto:	**64.863,68 €**	**64.863,68 €**	**46.164,58 €**	**46.164,58 €**	**18.699,10 €**	**28,83 %**
Rabatt 2%:		**-1.297,27 €**				
ENDSUMME Netto:	**63.566,41 €**	**63.566,41 €**	**46.164,58 €**	**46.164,58 €**	**17.401,83 €**	**27,38 %**

Abb. 26: Zusammenfassung Kalkulationsblatt

5.12 Umsetzung eines praktischen Beispiels

Als erstes Projekt wurde mit der Applikation ein Angebot einer schon laufenden Anlage nachkalkuliert. Hauptaugenmerk wurde dabei auf Genauigkeit, Arbeitsaufwand und Endsumme gelegt.

Bei dieser Anlage handelt es sich um eine HVAC (Heating Ventilation Air Conditioning) – Anlage für einen Kompostierungslaborraum auf Madeira. Für diesen Raum wurde eine Einzelraumregelung mithilfe einer DDC Regelung der Type "Excel" des Fabrikats Honeywell verwendet.

Eine Einzelraumregelung besteht im Normalfall aus einem Zu- und Abluftventilator, wo auch jeweils über Sensoren die Temperatur gemessen wird. Zusätzlich enthält die Zuluft einen Filter, welcher über einen Differenzdruckschalter auf Verschmutzung überwacht wird. Über ein Heizelement wird die Zuluft auf die gewünschte Temperatur gebracht. Die

Raumtemperatur wird über einen PI-Kaskadenregelkreis durch eine DDC geregelt. Zusätzliche Elemente, wie Keilriemenüberwachung, Außentemperatursensoren, Spannungsabfallmelder usw. vollenden die HVAC – Anlage. Zusätzlich besteht für die Ventilatoren und Heizregister die Möglichkeit den Automatikmodus zu verlassen und die Antriebe manuell über den Schaltschrank zu steuern. Dies ist für Wartungsarbeiten an der Anlage vorgesehen.
Die Erstellung des Angebotes von Hand aus war sehr aufwendig. Es mussten Produktkataloge und Preislisten durchforstet werden, um die richtigen Geräte ausfindig zu machen. Danach wurde die Regelung der Anlage definiert und die zugehörigen Datenpunkte und Module errechnet. Am Ende wurden Zuschläge und Rabatte kalkuliert. Alle diese Vorgänge mussten händisch durchgeführt werden, wodurch ein großer Zeitaufwand nötig war.

Mit der neuen Applikation hat sich das wesentlich vereinfacht. Da es eine Datenbank mit allen Geräten und Modulen gibt, können diese einfach über Auswahlboxen im Programm ausgewählt werden. Wurde ein Gerät gewählt, so werden alle Daten aus der Datenbank geladen, angefangen vom Einkaufspreis, bis zur genauen Typenbezeichnung und dem Ausschreibungstext. Nachdem alle Geräte definiert wurden, kann man noch aus einer Vielzahl von DDC Geräten auswählen, wodurch automatisch die benötigen zugehörigen Module berechnet werden. Sodann erfolgt die Eingabe der Reisekosten und über die Kalkulation erfährt der Benutzer nun die Endsumme. Er hat nun noch die Möglichkeit mit Rabatten und/oder Zuschlägen, das Angebot konkurrenzfähig auszulegen. Die Ausgabe des fertigen Angebots in Excel vollendet die Angebotserstellung.

Beim Vergleich zwischen der händischen Berechnung und der jetzigen Bearbeitung über die neue Applikation wurde festgestellt, dass der Arbeitsaufwand sehr vereinfacht und beschleunigt wurde. Das nachkalkulierte Angebot war vom Umfang her identisch zu dem bestehenden Projekt, einzig die Preise waren ein klein wenig unterschiedlich. Dies konnte dadurch erklärt werden, da das Projekt mit den aktuellen Preisen aus der Datenbank kalkuliert hat (siehe Tabelle 3).

Fühler und Geber	€ 199,-	€ 377,45
DDC-Regelung	€ 1644,-	€ 1859,-
Dokumentation	€ 620,-	€ 630,-
Schaltschrank	€ 3649,-	€ 4013,90
Inbetriebnahme	€ 2100,-	€ 2286,40
Gesamtpreis	**€ 8212,-**	**€ 9166,75**

Tab. 3: Vergleich Summen händische Berechnung – Kalkulation über die Anwendung

Die Vorteile des Programms sind:

- hohe Bearbeitungsgeschwindigkeit
- gute Benutzerfreundlichkeit
- automatische Kalkulation des gesamten Angebotes mit Ausgabe aller relevanten Daten

6 Zusammenfassung

6.1 Ergebnisse

Ziel dieser Diplomarbeit war es, ein Programm zu entwickeln, welches die Angebotslegung elektronisch unterstützen soll. Dabei wurde im Speziellen auf mess-, steuer- und regelungstechnische Anlagen, wie sie im Klimaanlagenbau Verwendung finden, eingegangen. Hauptaugenmerk wurde darauf gelegt, dass dem Mitarbeiter ein Großteil der händischen Arbeit bei einer Angebotserstellung abgenommen wird und viele Bearbeitungsschritte automatisiert wurden.

Es wurde ermöglicht in kürzerer Zeit mehrere Angebote zu erstellen, um dadurch auch die Auftragswahrscheinlichkeit zu erhöhen. Ein einheitliches, attraktives äußeres Erscheinungsbild vollendet das professionelle Angebot.

Durch diverse Überwachungen während der Kalkulationsphase eines Angebots mit dem Kalkulationsprogramm verringert sich automatisch die Fehlerhäufigkeit. Gleichzeitig wird das Angebotsrisiko durch z.B. zu geringe Aufschläge auf Nettopreise bei händischer Berechnung minimiert.

Durch die Möglichkeit, alte Projekte als Grundlage für ein neues Angebot zu verwenden, wird auch die Bearbeitung vereinfacht. Aber auch die Exportfunktion in das Programm Excel hat sich als sehr nützlich herausgestellt.

Aufgrund des immer größer werdenden, auch internationalen Wettbewerbs, sind als Entscheidungskriterien neben Qualität die auftretenden Kosten entscheidend. Da ein nur sehr geringer Auftragseingang im Vergleich zu einer hohen Angebotsabgabe besteht, muss auf sehr hohe Genauigkeit mit geringem Arbeitsaufwand in kurzer Zeit geachtet werden.
Diese Kriterien wurden mit diesem Programm vollends erfüllt, wobei in diesem auch noch viel Erweiterungspotential für die Zukunft vorhanden ist.

Die Anforderungen an das Programm wurden erfüllt und mittlerweile findet diese Applikation bei der Firma GAA bereits Verwendung.

6.2 Ausblick

Hat sich das Programm bei der Firma GAA über einen längeren Zeitraum bewährt, so könnte dies in Zukunft auch bei ihren Zweigstellen, wie zum Beispiel in Russland, eingesetzt werden. Dazu würde nur die Übersetzung der Textdateien ins Russische notwendig sein. Daraufhin könnte diese Applikation nach einer Einschulung des Personals sofort auch dort eingesetzt werden.

Die Applikation hat aber noch Erweiterungspotential. Durch Hinzufügen von weiteren Daten könnten Stücklisten, Projektpläne, Inbetriebnahmepläne und schließlich auch Teile der Dokumentation automatisch erstellt werden. Ein übergreifender Informationsfluss über einzelne Abschnitte eines Projektes wäre die Folge. Natürlich müssten diese Möglichkeiten auf Machbarkeit überprüft und eine Nutzwertanalyse der Firma GAA durchgeführt werden.

Literaturverzeichnis

(1) Beichter, Siegfried: Rechnergestützte Technische Problemlösung bei der Angebotserstellung von Flexiblen Drehzellen, Universität Karlsruhe, 1992

(2) Schwetz, Dipl.-Betriebsw. W: Computerunterstützte Angebotserstellung im Rahmen integrierter Vertriebssteuerungssysteme, 1990, Pfinztal, S136

(3) Kirsten, Dr. J.: Strategische Bedeutung von Bertriebsinformationsunterstützung, Wiesbaden, 1990

(4) Feller Achim H.: Kalkulation in der Angebotsphase mit dem selbsttätig abgeleiteten Erfahrungswissen der Arbeitsplanung, 1992, S2 f

(5) Feller Achim H.: Kalkulation in der Angebotsphase mit dem selbsttätig abgeleiteten Erfahrungswissen der Arbeitsplanung, 1992, S3

(6) Kirsten, Dr. J.: Strategische Bedeutung von Bertriebsinformationsunterstützung, Wiesbaden, 1990

(7) Hölzler, Dipl.-Ing. E: Kraus, Dipl.-Ing. H.: Konfiguration von System-Produkten – von der Anforderung zu Angebot und Stückliste, München, 1990, S66

(8) Richter Dipl.-Wirtsch.-Ing. R.: Stucky, Prof. Dr. rer. Nat. W: Datenbanksysteme im Angebotsbereich – Anforderungen und Trends, Karlsruhe, 1990, S181

(9) Schwetz, Dipl.Betriebsw. W.: Computerunterstützte Angebotserstellung im Rahmen integrierter Vertriebssteuerungssysteme, Pfinztal, 1990

(10) URL: http://dn.codegear.com/article/20781, verfügbar am 13.02.2009

(11) Barillo, Jim: URL: http://phx.corporate-ir.net/phoenix.zhtml?c=112793&p=irol-newsArticle&ID=792562&highlight=, verfügbar am 13.02.2009

(12) Smith, Ben: URL: http://dn.codegear.com/article/33818, verfügbar am 13.02.2009

(13) URL: http://entwickler.de/zonen/portale/psecom,id,214,news,32493,p,0.html, verfügbar am 13.02.2009

(14) URL: http://www.codegear.com/article/36464, verfügbar am 13.02.2009

(15) URL: http://www.codegear.com/products/cppbuilder, verfügbar am 13.02.2009

(16) Borland C++ Online Hilfe

(17) Schumann, Hans-Georg: C++ leicht & verständlich, 2. Aufl., Knowware, 2007

(18) Goll Joachim: Dausmann, Manfred: Bröckl, Ulrich: C als erste Programmiersprache, 5. Aufl. Teubner, Wiesbaden, 2005

(19) Schumann, Hans-Georg: C++ leicht & verständlich, 2. Aufl., Knowware, 2007

(20) ProBiq, Borland C++ Builder 5.0 Lehrbuch

(21) URL: http://www-aix.gsi.de/~giese/swr/ursache1.html, verfügbar am 01.03.2009

(22) Corsten, H: Software-Architekturen für das E-Business, 2000, S. 177.

(23) Kaiser, Prof. Richard, C++ mit dem Borland C++Builder 2007, 2. Auflage, Tübingen, 2008

(24) Borland C++ Online Hilfe

(25) Russell, Jordan: URL: www.jrsoftware.org, verfügbar am 10.02.2009

(26) Christof Hübner: Herrmann Merz: Thomas Hansemann; Gebäudeautomation: Kommunikationssysteme mit EIB/KNX, LON und BACnet, München, 2007

(27) URL: http://homepage.swissonline.ch/inux-GmbH/, verfügbar am 09.04.2009

Anhangsverzeichnis

Header-Datei Hauptprogramm:

```
//---------------------------------------------------------------------------

#ifndef Angebot_UH
#define Angebot_UH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Menus.hpp>
#include <ComCtrls.hpp>
#include <Dialogs.hpp>
#include <ExtCtrls.hpp>
#include <Db.hpp>
#include <DBTables.hpp>
#include <Grids.hpp>
#include <DBGrids.hpp>
#include <OleServer.hpp>
#include "Excel_2K_SRVR.h"
#include <Mask.hpp>
#include <DBCtrls.hpp>
#include <Buttons.hpp>
#include <ImgList.hpp>

//Structurdefinition

//---------------------------------------------------------------------------
class TF_Main : public TForm
{
__published:       // IDE-managed Components
        TMainMenu *MainMenu1;
        TMenuItem *Projekt1;
        TMenuItem *Exit1;
        TMenuItem *Bearbeiten1;
        TTreeView *TreeView1;
        TOpenDialog *OpenDialog1;
        TSaveDialog *SaveDialog1;
        TStatusBar *StatusBar1;
        TMenuItem *Info1;
        TMenuItem *Version1;
        TMenuItem *Sprache1;
        TMenuItem *Deutsch1;
        TMenuItem *Englisch1;
        TTimer *Timer1;
        TPageControl *PageControl1;
        TTabSheet *TabSheet1;
        TEdit *Edit_Nr;
        TEdit *Edit_Bear;
        TEdit *Edit_Ver;
        TEdit *Edit_AN;
        TEdit *Edit_PrjName;
        TEdit *Edit_Datum;
        TEdit *Edit_UZei;
        TEdit *Edit_Firma;
        TEdit *Edit_zHd;
        TEdit *Edit_Strasse;
        TEdit *Edit_Ort;
        TTable *Table1;
```

```
TDataSource *DataSource1;
TQuery *Query1;
TDatabase *Database1;
TStringGrid *StringGrid1;
TPanel *Panel1;
TLabel *Label1;
TPanel *Panel2;
TDBNavigator *DBNavigator1;
TDBMemo *DBMemo1;
TDBGrid *DBGrid1;
TDBEdit *DBEdit1;
TDBEdit *DBEdit2;
TDBEdit *DBEdit3;
TDBEdit *DBEdit4;
TDBEdit *DBEdit5;
TDBEdit *DBEdit6;
TDBEdit *DBEdit7;
TDBEdit *DBEdit8;
TDBEdit *DBEdit9;
TDBEdit *DBEdit10;
TDBEdit *DBEdit11;
TDBEdit *DBEdit12;
TDBEdit *DBEdit13;
TDBEdit *DBEdit14;
TDBEdit *DBEdit15;
TDBEdit *DBEdit16;
TLabel *Label2;
TLabel *Label3;
TLabel *Label4;
TLabel *Label5;
TLabel *Label6;
TLabel *Label7;
TLabel *Label8;
TLabel *Label9;
TLabel *Label10;
TLabel *Label11;
TLabel *Label12;
TLabel *Label13;
TLabel *Label14;
TLabel *Label15;
TLabel *Label16;
TLabel *Label17;
TLabel *Label18;
TButton *Button1;
TButton *Button2;
TPanel *Panel3;
TPageControl *PageControl2;
TPanel *Panel4;
TListView *ListView1;
TButton *Button4;
TButton *Button5;
TButton *Button6;
TLabel *Label19;
TLabel *Label20;
TDBEdit *DBEdit17;
TDBEdit *DBEdit18;
TDBEdit *DBEdit19;
TLabel *Label21;
TLabel *Label22;
TLabel *Label23;
```

```
TLabel *Label24;
TDBEdit *DBEdit20;
TDBEdit *DBEdit21;
TDBEdit *DBEdit22;
TProgressBar *ProgressBar1;
TGroupBox *GroupBox1;
TCheckBox *CheckBox1;
TEdit *Edit_Baut1;
TCheckBox *CheckBox2;
TCheckBox *CheckBox3;
TCheckBox *CheckBox4;
TCheckBox *CheckBox5;
TCheckBox *CheckBox6;
TEdit *Edit_Baut2;
TEdit *Edit_Baut3;
TEdit *Edit_Baut4;
TEdit *Edit_Baut5;
TEdit *Edit_Baut6;
TButton *Button7;
TCheckBox *CheckBox7;
TCheckBox *CheckBox8;
TEdit *Edit_Baut7;
TEdit *Edit_Baut8;
TTabSheet *TabSheet_Baut1;
TTabSheet *TabSheet_Baut2;
TTabSheet *TabSheet_Baut3;
TTabSheet *TabSheet_Baut4;
TTabSheet *TabSheet_Baut5;
TTabSheet *TabSheet_Baut6;
TTabSheet *TabSheet_Baut7;
TTabSheet *TabSheet_Baut8;
TRadioGroup *RadioGroup1;
TPopupMenu *PopupMenu1;
TMenuItem *Kopieren1;
TMenuItem *Einfgen1;
TCheckBox *CheckBox9;
TCheckBox *CheckBox10;
TEdit *Edit_Baut9;
TEdit *Edit_Baut10;
TTabSheet *TabSheet_Baut9;
TTabSheet *TabSheet_Baut10;
TPanel *Panel5;
TLabel *Label25;
TLabel *Label26;
TButton *Reset1;
TStringGrid *StringGrid_EK;
TStringGrid *StringGrid_VK;
TLabel *Label27;
TLabel *Label28;
TLabel *Label29;
TLabel *Label30;
TEdit *Edit_SW_EK;
TEdit *Edit_SW_VK;
TButton *Rabatt_EK_Speichern;
TButton *Neue_Zeile_Rabatt_EK;
TButton *Zeile_Loeschen_Rabatt_EK;
TButton *Neue_Zeile_Rabatt_VK;
TButton *Zeile_Loeschen_Rabatt_VK;
TButton *Datenuebernahme;
TTabSheet *TabSheet_GLT;
```

```
TTabSheet *TabSheet_SW;
TTabSheet *TabSheet_Gesamt;
TStringGrid *StringGrid_GLT;
TLabel *Label31;
TLabel *Label32;
TLabel *Label33;
TLabel *Label34;
TEdit *SW_Euro;
TLabel *Label35;
TLabel *Label36;
TComboBox *ComboBox1;
TComboBox *Anz_Reise_Euro;
TLabel *Label37;
TEdit *Reise_Euro;
TLabel *Label38;
TLabel *Label39;
TComboBox *Anz_Hotel_Euro;
TLabel *Label40;
TEdit *Hotel_Euro;
TLabel *Label41;
TLabel *Label42;
TEdit *SW_IBS_Ges_Euro;
TLabel *Label43;
TButton *Kalkulation;
TEdit *Edit_Anlagenname;
TEdit *Edit_Bezeichnung;
TButton *Button_ExportLV;
TButton *Button_ImportLV;
TPanel *Panel6;
TLabel *Label44;
TLabel *Label54;
TLabel *Label55;
TLabel *Label56;
TLabel *Label57;
TLabel *Label58;
TLabel *Label59;
TLabel *Label60;
TLabel *Label61;
TLabel *Label62;
TLabel *Label63;
TLabel *Label64;
TLabel *Label65;
TLabel *Label66;
TDBNavigator *DBNavigator2;
TDBMemo *DBMemo2;
TDBEdit *DBEdit23;
TDBEdit *DBEdit33;
TDBEdit *DBEdit34;
TDBEdit *DBEdit35;
TDBEdit *DBEdit36;
TDBEdit *DBEdit37;
TDBEdit *DBEdit38;
TButton *Button_Exp_DB2;
TButton *Button_Imp_DB2;
TDBEdit *DBEdit39;
TDBEdit *DBEdit40;
TDBEdit *DBEdit41;
TDBEdit *DBEdit42;
TDBEdit *DBEdit43;
TDBEdit *DBEdit44;
```

```
TProgressBar *ProgressBar_DB2;
TDataSource *DataSource2;
TTable *Table2;
TDatabase *Database2;
TQuery *Query2;
TDBGrid *DBGrid2;
TLabel *Label45;
TLabel *Label46;
TSpeedButton *SpeedButton_Save;
TSpeedButton *SpeedButton_Open;
TSpeedButton *SpeedButton_K;
TSpeedButton *SpeedButton_LV;
TImageList *ImageList1;
TRichEdit *RichEdit_Gesamt;
TDBCheckBox *DBCheckBox1;
TDBEdit *DBEdit24;
TLabel *Label47;
TPanel *OptionenPanel;
TLabel *Label48;
TComboBox *ComboBox_DDC_Option;
TLabel *Label49;
TLabel *Label50;
TEdit *SW_IBS_Ges_Euro_EK;
TLabel *Label51;
TLabel *Label52;
TEdit *SW_Euro_EK;
TLabel *Label53;
TUpDown *UpDown1;
TMaskEdit *KundenRabatt;
TLabel *Label67;
TRadioGroup *RadioGroup_Preisauswahl;
TLabel *Label68;
TEdit *SW_Pro_Datenpkt;
TLabel *Label69;
TEdit *Doku_Pro_Datenpkt;
TLabel *Label70;
TEdit *SW_Pro_Datenpkt_EK;
TLabel *Label71;
TEdit *Doku_Pro_Datenpkt_EK;
TLabel *Label72;
TLabel *Label73;
TLabel *Label74;
TEdit *IBN_Pro_Datenpkt;
TLabel *Label75;
TEdit *IBN_Pro_Datenpkt_EK;
TLabel *Label76;
TLabel *Label77;
TEdit *Doku_Euro;
TLabel *Label78;
TEdit *Doku_Euro_EK;
TLabel *Label79;
TEdit *IBN_Euro;
TLabel *Label80;
TEdit *IBN_Euro_EK;
TLabel *Label81;
TEdit *Datenpunkte;
TTabSheet *TabSheet2;
TRichEdit *RichEdit_Kalkulationsblatt;
TEdit *Reise_Euro_EK;
TLabel *Label82;
```

```
    TEdit *Hotel_Euro_EK;
    TLabel *Label83;
    TEdit *Reisekosten_Euro;
    TEdit *Reisekosten_Euro_EK;
    TLabel *Label84;
    TLabel *Label85;
    TGroupBox *GroupBox2;
    TCheckBox *SubangebotSS;
    TEdit *SS_Subangebot_Euro;
    TEdit *SS_Aufschlag;
    TUpDown *UpDown2;
    TLabel *Label86;
    TLabel *Label87;
    TButton *Export_Gesamt;
    TButton *Export_Kalkulationsblatt;
    TLabel *Label88;
    void __fastcall Exit1Click(TObject *Sender);
    void __fastcall TreeView1DblClick(TObject *Sender);
    void __fastcall TreeView1Click(TObject *Sender);
    void __fastcall Version1Click(TObject *Sender);
    void __fastcall Deutsch1Click(TObject *Sender);
    void __fastcall Englisch1Click(TObject *Sender);
    void __fastcall FormCreate(TObject *Sender);
    void __fastcall Timer1Timer(TObject *Sender);
    void __fastcall StringGrid1Click(TObject *Sender);
    void __fastcall StringGrid1TopLeftChanged(TObject *Sender);
    void __fastcall FormResize(TObject *Sender);
    void __fastcall StringGrid1DrawCell(TObject *Sender, int ACol,
      int ARow, TRect &Rect, TGridDrawState State);
    void __fastcall Button1Click(TObject *Sender);
    void __fastcall Button2Click(TObject *Sender);
    void __fastcall StringGrid1ContextPopup(TObject *Sender,
      TPoint &MousePos, bool &Handled);
    void __fastcall Kopieren1Click(TObject *Sender);
    void __fastcall Einfgen1Click(TObject *Sender);
    void __fastcall Button4Click(TObject *Sender);
    void __fastcall Button5Click(TObject *Sender);
    void __fastcall Button6Click(TObject *Sender);
    void __fastcall Edit_FirmaChange(TObject *Sender);
    void __fastcall CheckBox1Click(TObject *Sender);
    void __fastcall CheckBox2Click(TObject *Sender);
    void __fastcall CheckBox3Click(TObject *Sender);
    void __fastcall CheckBox4Click(TObject *Sender);
    void __fastcall CheckBox5Click(TObject *Sender);
    void __fastcall CheckBox6Click(TObject *Sender);
    void __fastcall CheckBox7Click(TObject *Sender);
    void __fastcall CheckBox8Click(TObject *Sender);
    void __fastcall Button7Click(TObject *Sender);
    void __fastcall RadioGroup1Click(TObject *Sender);
    void __fastcall CheckBox9Click(TObject *Sender);
    void __fastcall CheckBox10Click(TObject *Sender);
    void __fastcall Reset1Click(TObject *Sender);
    void __fastcall Rabatt_EK_SpeichernClick(TObject *Sender);
    void __fastcall Neue_Zeile_Rabatt_EKClick(TObject *Sender);
    void __fastcall Zeile_Loeschen_Rabatt_EKClick(TObject *Sender);
    void __fastcall StringGrid_EKSelectCell(TObject *Sender, int ACol,
      int ARow, bool &CanSelect);
    void __fastcall Neue_Zeile_Rabatt_VKClick(TObject *Sender);
    void __fastcall Zeile_Loeschen_Rabatt_VKClick(TObject *Sender);
    void __fastcall StringGrid_VKSelectCell(TObject *Sender, int ACol,
```

```
        int ARow, bool &CanSelect);
      void __fastcall DatenuebernahmeClick(TObject *Sender);
      void __fastcall SW_EuroChange(TObject *Sender);
      void __fastcall KalkulationClick(TObject *Sender);
      void __fastcall Button_ExportLVClick(TObject *Sender);
      void __fastcall Button_ImportLVClick(TObject *Sender);
      void __fastcall Button_Exp_DB2Click(TObject *Sender);
      void __fastcall Button_Imp_DB2Click(TObject *Sender);
      void __fastcall SpeedButton_SaveClick(TObject *Sender);
      void __fastcall SpeedButton_OpenClick(TObject *Sender);
      void __fastcall SpeedButton_KClick(TObject *Sender);
      void __fastcall SpeedButton_LVClick(TObject *Sender);
      void __fastcall Reise_EuroChange(TObject *Sender);
      void __fastcall Hotel_EuroChange(TObject *Sender);
      void __fastcall Anz_Reise_EuroChange(TObject *Sender);
      void __fastcall Anz_Hotel_EuroChange(TObject *Sender);
      void __fastcall ComboBox_DDC_OptionChange(TObject *Sender);
      void __fastcall SW_Pro_DatenpktChange(TObject *Sender);
      void __fastcall SW_Pro_Datenpkt_EKChange(TObject *Sender);
      void __fastcall Doku_Pro_DatenpktChange(TObject *Sender);
      void __fastcall Doku_Pro_Datenpkt_EKChange(TObject *Sender);
      void __fastcall IBN_Pro_DatenpktChange(TObject *Sender);
      void __fastcall IBN_Pro_Datenpkt_EKChange(TObject *Sender);
      void __fastcall Reise_Euro_EKChange(TObject *Sender);
      void __fastcall Hotel_Euro_EKChange(TObject *Sender);
      void __fastcall SubangebotSSClick(TObject *Sender);
      void __fastcall SS_Subangebot_EuroChange(TObject *Sender);
      void __fastcall Export_GesamtClick(TObject *Sender);
private:  // User declarations
  void Sprache(bool);
  void Lesen(AnsiString);
  void Schreiben(AnsiString);
  void AufrufAdressver(void);
  void AufrufRabattblatt(void);
    //Zeiger auf das TTabSheet
  TTabSheet* pPage;
   // Zeiger auf die TComboBox für die Bearbeitung der Zelleninhalten
  TComboBox* pComboEditor;
   // Zeiger auf die TStringGrid für StringGrid Bauteil1
  TStringGrid* pStringGrid_Baut1;
  TStringGrid* pStringGrid_Baut2;
  TStringGrid* pStringGrid_Baut3;
  TStringGrid* pStringGrid_Baut4;
  TStringGrid* pStringGrid_Baut5;
  TStringGrid* pStringGrid_Baut6;
  TStringGrid* pStringGrid_Baut7;
  TStringGrid* pStringGrid_Baut8;
  TStringGrid* pStringGrid_Baut9;
  TStringGrid* pStringGrid_Baut10;
  TStringGrid* StrGd;
   // Positionierung der ComboBox in der ausgewählten Zelle
  void __fastcall PlaceComboEditor(void);
   // Eventhandler für die Änderung der Auswahl in der Inplace-ComboBox
  void __fastcall ComboEditorChange(TObject *Sender);
   // Füllen der ComboBox mit Grobauswahl
  void __fastcall LoadGrobToCombo(TComboBox *pComboBox);
   // Füllen der ComboBox mit den Fein1
  void __fastcall LoadFein1ToCombo(TComboBox *pComboBox, int ACol, int ARow);
   // Überprüfen auf vollständige Zeile
  void __fastcall CheckForDevice(int,TStringGrid *StrGd);
```

```
    // StringGrid GLT befüllen
  int __fastcall FillGLT(TStringGrid *StrGd, AnsiString Bauteil, int Index);
    // speichern in verdeckte StringGrids
  void __fastcall TF_Main::SaveToPStrGrid(void);
  void __fastcall TF_Main::AngebotKalk(TStringGrid *StrGd, AnsiString text1, float Preis[4]);
  void __fastcall TF_Main::AngebotKalkblatt(TStringGrid *StrGd, AnsiString text1);
  void __fastcall TF_Main::AngebotZusammenstel(float I, float II, float III, float IV);
  void __fastcall TF_Main::AngebotZusammenstelKalkblatt(AnsiString text, float PreisI[4], float PreisII[4], float
PreisIII[4], float PreisIV[4]);
  void __fastcall TF_Main::AngebotZusammenstelKalkblattAnzeige(AnsiString text, float Preis[4],bool fett);

public:                 // User declarations
       __fastcall TF_Main(TComponent* Owner);  //Bestehender Konstruktor
       __fastcall ~TF_Main();                  //Deklaration des Destruktors

      // Berechnung der Rabattprozente
      float __fastcall Rabkalk(AnsiString RabGr);
      float __fastcall RabkalkEK(AnsiString RabGr);
};
//---------------------------------------------------------------------------
extern PACKAGE TF_Main *F_Main;
//---------------------------------------------------------------------------
#endif
```

Quelldatei Hauptprogramm:

```
//---------------------------------------------------------------------------

#include <fstream>      //um datein zu öffnen
using namespace std;    //um datein zu öffnen
#include <vcl.h>

#pragma hdrstop

#include "AboutBox_U.h"
#include "EditiernNeuerEintrag_U.h"
#include "Abfrage_Oeffnen.h"
#include "TKalkulation.h"
#include "Angebot_U.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TF_Main *F_Main;

TKalkulation *pKalkulation_Baut1, *pKalkulation_Baut2;
TKalkulation *pKalkulation_Baut3, *pKalkulation_Baut4;
TKalkulation *pKalkulation_Baut5, *pKalkulation_Baut6;
TKalkulation *pKalkulation_Baut7, *pKalkulation_Baut8;
TKalkulation *pKalkulation_Baut9, *pKalkulation_Baut10;

//Konstantendeklaration
const char *Const_Version = "Version: V0.0.1.0";
const int AnzahlZeilen = 20;
const int AnzahlZeilenExpImp = 500;

//Globale Variablendeklartion
AnsiString Dir;
AnsiString Zelle1, Zelle2, Zelle3, Zelle4, Zelle5, Zelle6, Zelle7, Zelle8, Zelle9, Zelle10, Zelle11;
AnsiString History_SW_Euro = "0,00", History_Reise_Euro = "0,00",History_Reise_Euro_EK = "0,00",
History_Hotel_Euro = "0,00",History_Hotel_Euro_EK = "0,00",History_SW_Pro_Datenpkt=
"0,00",History_SW_Pro_Datenpkt_EK= "0,00";
AnsiString History_Doku_Pro_Datenpkt = "0,00", History_Doku_Pro_Datenpkt_EK = "0,00",
History_IBN_Pro_Datenpkt = "0,00", History_IBN_Pro_Datenpkt_EK = "0,00";
AnsiString History_SS_Subangebot_Euro = "0,00";
int copycol, copyrow, RadioButtonBautSav = -1, delRow_EK = 0, delRow_VK = 0;
bool AenderungDurchgefuhert = false, FMainCaption = false;
//---------------------------------------------------------------------------
// **************eigene Funktionen**************
//---------------------------------------------------------------------------

void __fastcall TF_Main::FormCreate(TObject *Sender)
{

Dir = GetCurrentDir();  //liefert aktuelles Verzeichnis
//aktuelle Version in Statusleiste anzeigen
StatusBar1->Panels->Items[0]->Text = Const_Version;
//aktuelles Verzeichnis in Statusleiste anzeigen
//StatusBar1->Panels->Items[1]->Text = Dir+ "\\";

  Table1->Active = false;
//Datein von Datenbank in Sav kopieren
//::CopyFile("D:\\BorlandC++\\Projekte\\Angebotslegungsprogramm\\DB\\Angebot_DB_Aktuell.DB","D:\\Borl
andC++\\Projekte\\Angebotslegungsprogramm\\DB\\Sav\\Angebot_DB_Aktuell.DB",false);
```

```
//::CopyFile("D:\\BorlandC++\\Projekte\\Angebotslegungsprogramm\\DB\\Angebot_DB_Aktuell.MB","D:\\Borl
andC++\\Projekte\\Angebotslegungsprogramm\\DB\\Sav\\Angebot_DB_Aktuell.MB",false);
//::CopyFile("D:\\BorlandC++\\Projekte\\Angebotslegungsprogramm\\DB\\Angebot_DB_Aktuell.PX","D:\\Borl
andC++\\Projekte\\Angebotslegungsprogramm\\DB\\Sav\\Angebot_DB_Aktuell.PX",false);

//::CopyFile(Dir+"\\DB\\Angebot_DB_Aktuell.DB",Dir+"\\DB\\Sav\\Angebot_DB_Aktuell.DB",false);
//::CopyFile(Dir+"\\DB\\Angebot_DB_Aktuell.MB",Dir+"\\DB\\Sav\\Angebot_DB_Aktuell.MB",false);
//::CopyFile(Dir+"\\DB\\Angebot_DB_Aktuell.PX",Dir+"\\DB\\Sav\\Angebot_DB_Aktuell.PX",false);

//Für ComboBoxes
 // ComboBox für die Bearbeteung der Einträge in
  // der 1. Spalte vorbereiten:
  pComboEditor = new TComboBox(this);
  pComboEditor->Parent = this;
  pComboEditor->Visible = false;
  pComboEditor->Style = Stdctrls::csDropDownList;
  pComboEditor->OnChange = ComboEditorChange;

  // Spaltenüberschriften eintragen:
  StringGrid1->Cells[0][0] = "OK";
  StringGrid1->Cells[1][0] = "Stück";
  StringGrid1->Cells[2][0] = "Grobauswahl";
  StringGrid1->Cells[3][0] = "Fein1";
  StringGrid1->Cells[4][0] = "Fein2";
  StringGrid1->Cells[5][0] = "Fein3";
  StringGrid1->Cells[6][0] = "Fein4";
  StringGrid1->Cells[7][0] = "Fein5";
  StringGrid1->Cells[8][0] = "Fein6";
  StringGrid1->Cells[9][0] = "Fein7";
  StringGrid1->Cells[10][0] = "Fein8";
  StringGrid1->Cells[11][0] = "Fein9";
  StringGrid1->Cells[12][0] = "Gerätetyp";
  StringGrid1->Cells[13][0] = "Gerätetyp2";
  StringGrid1->Cells[14][0] = "Gerätetyp3";
  StringGrid1->Cells[15][0] = "AE";
  StringGrid1->Cells[16][0] = "AA";
  StringGrid1->Cells[17][0] = "DE";
  StringGrid1->Cells[18][0] = "DA";
  StringGrid1->Cells[19][0] = "Hersteller";
  StringGrid1->Cells[20][0] = "RabattgruppeEK";
  StringGrid1->Cells[21][0] = "RabattgruppeVK";
  StringGrid1->Cells[22][0] = "ListenpreisType1";
  StringGrid1->Cells[23][0] = "ListenpreisType2";
  StringGrid1->Cells[24][0] = "ListenpreisType3";
  StringGrid1->Cells[25][0] = "BM-Gruppe";
  StringGrid1->Cells[26][0] = "Ausschreibungstext";

  // Tabelle mit Beispieldaten füllen
  StringGrid1->Cells[5][1] = TDateTime(Now()).FormatString("dd.mm.yyyy");

  pStringGrid_Baut1 = new TStringGrid(this);
  pStringGrid_Baut1->Parent = this;
  pStringGrid_Baut1->Visible = false;
  pStringGrid_Baut1->ColCount = 27;
  pStringGrid_Baut1->RowCount = 500;
  for(int i = 0; i<=26;i++)
   pStringGrid_Baut1->Cells[i][0]= StringGrid1->Cells[i][0];

  pStringGrid_Baut2 = new TStringGrid(this);
  pStringGrid_Baut2->Parent = this;
```

```
pStringGrid_Baut2->Visible = false;
pStringGrid_Baut2->ColCount = 27;
pStringGrid_Baut2->RowCount = 500;
for(int i = 0; i<=26;i++)
 pStringGrid_Baut2->Cells[i][0]= StringGrid1->Cells[i][0];

pStringGrid_Baut3 = new TStringGrid(this);
pStringGrid_Baut3->Parent = this;
pStringGrid_Baut3->Visible = false;
pStringGrid_Baut3->ColCount = 27;
pStringGrid_Baut3->RowCount = 500;
for(int i = 0; i<=26;i++)
 pStringGrid_Baut3->Cells[i][0]= StringGrid1->Cells[i][0];

pStringGrid_Baut4 = new TStringGrid(this);
pStringGrid_Baut4->Parent = this;
pStringGrid_Baut4->Visible = false;
pStringGrid_Baut4->ColCount = 27;
pStringGrid_Baut4->RowCount = 500;
for(int i = 0; i<=26;i++)
 pStringGrid_Baut4->Cells[i][0]= StringGrid1->Cells[i][0];

pStringGrid_Baut5 = new TStringGrid(this);
pStringGrid_Baut5->Parent = this;
pStringGrid_Baut5->Visible = false;
pStringGrid_Baut5->ColCount = 27;
pStringGrid_Baut5->RowCount = 500;
for(int i = 0; i<=26;i++)
 pStringGrid_Baut5->Cells[i][0]= StringGrid1->Cells[i][0];

pStringGrid_Baut6 = new TStringGrid(this);
pStringGrid_Baut6->Parent = this;
pStringGrid_Baut6->Visible = false;
pStringGrid_Baut6->ColCount = 27;
pStringGrid_Baut6->RowCount = 500;
for(int i = 0; i<=26;i++)
 pStringGrid_Baut6->Cells[i][0]= StringGrid1->Cells[i][0];

pStringGrid_Baut7 = new TStringGrid(this);
pStringGrid_Baut7->Parent = this;
pStringGrid_Baut7->Visible = false;
pStringGrid_Baut7->ColCount = 27;
pStringGrid_Baut7->RowCount = 500;
for(int i = 0; i<=26;i++)
 pStringGrid_Baut7->Cells[i][0]= StringGrid1->Cells[i][0];

pStringGrid_Baut8 = new TStringGrid(this);
pStringGrid_Baut8->Parent = this;
pStringGrid_Baut8->Visible = false;
pStringGrid_Baut8->ColCount = 27;
pStringGrid_Baut8->RowCount = 500;
for(int i = 0; i<=26;i++)
 pStringGrid_Baut8->Cells[i][0]= StringGrid1->Cells[i][0];

pStringGrid_Baut9 = new TStringGrid(this);
pStringGrid_Baut9->Parent = this;
pStringGrid_Baut9->Visible = false;
pStringGrid_Baut9->ColCount = 27;
pStringGrid_Baut9->RowCount = 500;
for(int i = 0; i<=26;i++)
```

```
   pStringGrid_Baut9->Cells[i][0]= StringGrid1->Cells[i][0];

  pStringGrid_Baut10 = new TStringGrid(this);
  pStringGrid_Baut10->Parent = this;
  pStringGrid_Baut10->Visible = false;
  pStringGrid_Baut10->ColCount = 27;
  pStringGrid_Baut10->RowCount = 500;
  for(int i = 0; i<=26;i++)
   pStringGrid_Baut10->Cells[i][0]= StringGrid1->Cells[i][0];

  StrGd = new TStringGrid(this);

  //Rabattblatt kreieren!!!!

  StringGrid_EK->Cells[0][0] = "Rabattgruppe";
  StringGrid_EK->Cells[1][0] = "Rabatt[%]";
  StringGrid_EK->Cells[2][0] = "Bemerkung";

  //Laden aus Datei fix
  TStringList* myList = new TStringList();
  myList->LoadFromFile(Dir+"\\files\\Rabatt.blatt");

  StringGrid_EK->RowCount = myList->Count/3+1;
  Edit_SW_EK->Text = myList->Strings[0];
  for (int i = 0; i < (myList->Count/3); i++)
   {
    StringGrid_EK->Cells[0][i+1] = myList->Strings[1+i*3];
    StringGrid_EK->Cells[1][i+1] = myList->Strings[2+i*3];
    StringGrid_EK->Cells[2][i+1] = myList->Strings[3+i*3];
   }

//Rabattgruppe aus Projektdatei

  StringGrid_VK->Cells[0][0] = "Rabattgruppe";
  StringGrid_VK->Cells[1][0] = "Rabatt[%]";
  StringGrid_VK->Cells[2][0] = "Bemerkung";

  StringGrid_VK->RowCount = myList->Count/3+1;
  Edit_SW_VK->Text = myList->Strings[0];
  for (int i = 0; i < (myList->Count/3); i++)
   {
    StringGrid_VK->Cells[0][i+1] = myList->Strings[1+i*3];
    StringGrid_VK->Cells[1][i+1] = myList->Strings[2+i*3];
    StringGrid_VK->Cells[2][i+1] = myList->Strings[3+i*3];
   }
delete myList;

//Erstellen der Kalkulation
pKalkulation_Baut1 = new TKalkulation(TabSheet_Baut1);
pKalkulation_Baut1->Parent = TabSheet_Baut1;
pKalkulation_Baut1->Visible = true;

pKalkulation_Baut2 = new TKalkulation(TabSheet_Baut2);
pKalkulation_Baut2->Parent = TabSheet_Baut2;
pKalkulation_Baut2->Visible = true;

pKalkulation_Baut3 = new TKalkulation(TabSheet_Baut3);
pKalkulation_Baut3->Parent = TabSheet_Baut3;
pKalkulation_Baut3->Visible = true;
```

```
pKalkulation_Baut4 = new TKalkulation(TabSheet_Baut4);
pKalkulation_Baut4->Parent = TabSheet_Baut4;
pKalkulation_Baut4->Visible = true;

pKalkulation_Baut5 = new TKalkulation(TabSheet_Baut5);
pKalkulation_Baut5->Parent = TabSheet_Baut5;
pKalkulation_Baut5->Visible = true;

pKalkulation_Baut6 = new TKalkulation(TabSheet_Baut6);
pKalkulation_Baut6->Parent = TabSheet_Baut6;
pKalkulation_Baut6->Visible = true;

pKalkulation_Baut7 = new TKalkulation(TabSheet_Baut7);
pKalkulation_Baut7->Parent = TabSheet_Baut7;
pKalkulation_Baut7->Visible = true;

pKalkulation_Baut8 = new TKalkulation(TabSheet_Baut8);
pKalkulation_Baut8->Parent = TabSheet_Baut8;
pKalkulation_Baut8->Visible = true;

pKalkulation_Baut9 = new TKalkulation(TabSheet_Baut9);
pKalkulation_Baut9->Parent = TabSheet_Baut9;
pKalkulation_Baut9->Visible = true;

pKalkulation_Baut10 = new TKalkulation(TabSheet_Baut10);
pKalkulation_Baut10->Parent = TabSheet_Baut10;
pKalkulation_Baut10->Visible = true;


StringGrid_GLT->Cells[0][0] = "Bauteil";
StringGrid_GLT->Cells[1][0] = "Stück";
StringGrid_GLT->Cells[2][0] = "Gerät";
StringGrid_GLT->Cells[3][0] = "Gerätetyp";
StringGrid_GLT->Cells[4][0] = "Gerätetyp2";
StringGrid_GLT->Cells[5][0] = "Gerätetyp3";
StringGrid_GLT->Cells[6][0] = "Hersteller";
StringGrid_GLT->Cells[7][0] = "RabattgruppeEK";
StringGrid_GLT->Cells[8][0] = "RabattgruppeVK";
StringGrid_GLT->Cells[9][0] = "ListenpreisType1";
StringGrid_GLT->Cells[10][0] = "ListenpreisType2";
StringGrid_GLT->Cells[11][0] = "ListenpreisType3";
}

//---------------------------------------------------------------------------
// Auswahl sprache
//---------------------------------------------------------------------------
void TF_Main::Sprache(bool a)
{
//Umschaltung Englisch - Deutsch
TStringList* myList = new TStringList();
if (a)
  myList->LoadFromFile(Dir+"\\txt\\captions_de.txt");
  else myList->LoadFromFile(Dir+"\\txt\\captions_gb.txt");

MainMenu1->Items->Items[0]->Caption = myList->Strings[0].SubString(9,15);
MainMenu1->Items->Items[1]->Caption = myList->Strings[10].SubString(9,15);
MainMenu1->Items->Items[1]->Items[0]->Caption = myList->Strings[11].SubString(9,15);
MainMenu1->Items->Items[1]->Items[0]->Items[0]->Caption = myList->Strings[12].SubString(9,15);
MainMenu1->Items->Items[1]->Items[0]->Items[1]->Caption = myList->Strings[13].SubString(9,15);
MainMenu1->Items->Items[2]->Caption = myList->Strings[20].SubString(9,15);
```

```
MainMenu1->Items->Items[2]->Items[0]->Caption = myList->Strings[21].SubString(9,15);
MainMenu1->Items->Items[3]->Caption = myList->Strings[30].SubString(9,15);
delete myList;

TStringList* myList1 = new TStringList();
if (a)
  myList1->LoadFromFile(Dir+"\\txt\\captions_Baum_de.txt");
  else myList1->LoadFromFile(Dir+"\\txt\\captions_Baum_gb.txt");

TreeView1->Items->Item[0]->Text = myList1->Strings[0].SubString(9,40);
TreeView1->Items->Item[1]->Text = myList1->Strings[1].SubString(9,40);
TreeView1->Items->Item[2]->Text = myList1->Strings[2].SubString(9,40);
TreeView1->Items->Item[3]->Text = myList1->Strings[3].SubString(9,40);
TreeView1->Items->Item[4]->Text = myList1->Strings[4].SubString(9,40);
TreeView1->Items->Item[5]->Text = myList1->Strings[5].SubString(9,40);
TreeView1->Items->Item[6]->Text = myList1->Strings[6].SubString(9,40);
TreeView1->Items->Item[7]->Text = myList1->Strings[7].SubString(9,40);
TreeView1->Items->Item[8]->Text = myList1->Strings[8].SubString(9,40);
TreeView1->Items->Item[9]->Text = myList1->Strings[10].SubString(9,40);
TreeView1->Items->Item[10]->Text = myList1->Strings[11].SubString(9,40);
TreeView1->Items->Item[11]->Text = myList1->Strings[12].SubString(9,40);
TreeView1->Items->Item[12]->Text = myList1->Strings[13].SubString(9,40);
TreeView1->Items->Item[13]->Text = myList1->Strings[20].SubString(9,40);
TreeView1->Items->Item[14]->Text = myList1->Strings[21].SubString(9,40);
TreeView1->Items->Item[15]->Text = myList1->Strings[22].SubString(9,40);
TreeView1->Items->Item[16]->Text = myList1->Strings[30].SubString(9,40);
TreeView1->Items->Item[17]->Text = myList1->Strings[31].SubString(9,40);
TreeView1->Items->Item[18]->Text = myList1->Strings[32].SubString(9,40);
TreeView1->Items->Item[19]->Text = myList1->Strings[40].SubString(9,40);
TreeView1->Items->Item[20]->Text = myList1->Strings[41].SubString(9,40);
TreeView1->Items->Item[21]->Text = myList1->Strings[42].SubString(9,40);
delete myList1;
}
//---------------------------------------------------------------------------
// Projekt speichern
//---------------------------------------------------------------------------
void TF_Main::Schreiben(AnsiString PrjFile)
{
TProgressBar *p = new TProgressBar(StatusBar1);
p->Parent = StatusBar1;
p->Top = 2;
p->Left = StatusBar1->Panels->Items[0]->Width + 2;
p->Width = StatusBar1->Panels->Items[1]->Width -2;
p->Height = StatusBar1->Height-2;
p->Smooth = true;
p->Step = 1;
p->Min = 0;
p->Max = 11;
StatusBar1->Repaint();

 TStringList* myList = new TStringList();

 p->StepIt();
 myList->Add(Edit_Firma->Text);
 myList->Add(Edit_zHd->Text);
 myList->Add(Edit_Strasse->Text);
 myList->Add(Edit_Ort->Text);
 myList->Add(Edit_Nr->Text);
 myList->Add(Edit_Ver->Text);
 myList->Add(Edit_AN->Text);
```

```
 myList->Add(Edit_PrjName->Text);
 myList->Add(Edit_Bear->Text);
 myList->Add(Edit_UZei->Text);
 myList->Add(Edit_Datum->Text);
 myList->Add(Edit_Anlagenname->Text);
 myList->Add(Edit_Bezeichnung->Text);

 myList->Add("");
 myList->Add("");

 if (CheckBox1->Checked) myList->Add(1);
 else myList->Add(0);
 myList->Add(Edit_Baut1->Text);
 if (CheckBox2->Checked) myList->Add(1);
 else myList->Add(0);
 myList->Add(Edit_Baut2->Text);
 if (CheckBox3->Checked) myList->Add(1);
 else myList->Add(0);
 myList->Add(Edit_Baut3->Text);
 if (CheckBox4->Checked) myList->Add(1);
 else myList->Add(0);
 myList->Add(Edit_Baut4->Text);
 if (CheckBox5->Checked) myList->Add(1);
 else myList->Add(0);
 myList->Add(Edit_Baut5->Text);
 if (CheckBox6->Checked) myList->Add(1);
 else myList->Add(0);
 myList->Add(Edit_Baut6->Text);
 if (CheckBox7->Checked) myList->Add(1);
 else myList->Add(0);
 myList->Add(Edit_Baut7->Text);
 if (CheckBox8->Checked) myList->Add(1);
 else myList->Add(0);
 myList->Add(Edit_Baut8->Text);
 if (CheckBox9->Checked) myList->Add(1);
 else myList->Add(0);
 myList->Add(Edit_Baut9->Text);
 if (CheckBox10->Checked) myList->Add(1);
 else myList->Add(0);
 myList->Add(Edit_Baut10->Text);

 myList->Add("");
 myList->Add("");
 myList->Add("");
 p->StepIt();
for (int ii = 1; ii <= 10; ii++)
 {
 for (int i = 1; i <= AnzahlZeilen; i++)
   for (int j = 1; j <= 11; j++)
    switch (ii)
    {
    case  1 : myList->Add(pStringGrid_Baut1->Cells[j][i]);
              break;
    case  2 : myList->Add(pStringGrid_Baut2->Cells[j][i]);
              break;
    case  3 : myList->Add(pStringGrid_Baut3->Cells[j][i]);
              break;
    case  4 : myList->Add(pStringGrid_Baut4->Cells[j][i]);
              break;
    case  5 : myList->Add(pStringGrid_Baut5->Cells[j][i]);
```

```
            break;
    case  6 : myList->Add(pStringGrid_Baut6->Cells[j][i]);
            break;
    case  7 : myList->Add(pStringGrid_Baut7->Cells[j][i]);
            break;
    case  8 : myList->Add(pStringGrid_Baut8->Cells[j][i]);
            break;
    case  9 : myList->Add(pStringGrid_Baut9->Cells[j][i]);
            break;
    case  10: myList->Add(pStringGrid_Baut10->Cells[j][i]);
            break;
    };

 p->StepIt();
 }
  myList->Add(Edit_SW_VK->Text);
  for (int h=1; h < StringGrid_VK->RowCount; h++)
   {
    myList->Add(StringGrid_VK->Cells[0][h]);
    myList->Add(StringGrid_VK->Cells[1][h]);
    myList->Add(StringGrid_VK->Cells[2][h]);
   }

 myList->SaveToFile(PrjFile);
 p->Position = 0;
 delete myList;
 delete p;
}
//---------------------------------------------------------------------------
// Projekt laden
//---------------------------------------------------------------------------
void TF_Main::Lesen(AnsiString PrjFile)
{
TProgressBar *p = new TProgressBar(StatusBar1);
p->Parent = StatusBar1;
p->Top = 2;
p->Left = StatusBar1->Panels->Items[0]->Width + 2;
p->Width = StatusBar1->Panels->Items[1]->Width -2;
p->Height = StatusBar1->Height-2;
p->Smooth = true;
p->Step = 1;
p->Min = 0;
p->Max = 11;
StatusBar1->Repaint();

 TStringList* myList = new TStringList();
 myList->LoadFromFile(PrjFile);

 int MyListCount = 0;
 MyListCount = myList->Count;

 p->StepIt();
 Edit_Firma->Text = myList->Strings[0];
 Edit_zHd->Text = myList->Strings[1];
 Edit_Strasse->Text = myList->Strings[2];
 Edit_Ort->Text = myList->Strings[3];
 Edit_Nr->Text = myList->Strings[4];
 Edit_Ver->Text = myList->Strings[5];
 Edit_AN->Text = myList->Strings[6];
 Edit_PrjName->Text = myList->Strings[7];
```

```
Edit_Bear->Text = myList->Strings[8];
Edit_UZei->Text = myList->Strings[9];
Edit_Datum->Text = myList->Strings[10];
Edit_Anlagenname->Text = myList->Strings[11];
Edit_Bezeichnung->Text = myList->Strings[12];
//= myList->Strings[13];
//= myList->Strings[14];

if (myList->Strings[15] == 1) CheckBox1->Checked = true;
else CheckBox1->Checked = false;
Edit_Baut1->Text = myList->Strings[16];
if (myList->Strings[17] == 1) CheckBox2->Checked = true;
else CheckBox2->Checked = false;
Edit_Baut2->Text = myList->Strings[18];
if (myList->Strings[19] == 1) CheckBox3->Checked = true;
else CheckBox3->Checked = false;
Edit_Baut3->Text = myList->Strings[20];
if (myList->Strings[21] == 1) CheckBox4->Checked = true;
else CheckBox4->Checked = false;
Edit_Baut4->Text = myList->Strings[22];
if (myList->Strings[23] == 1) CheckBox5->Checked = true;
else CheckBox5->Checked = false;
Edit_Baut5->Text = myList->Strings[24];
if (myList->Strings[25] == 1) CheckBox6->Checked = true;
else CheckBox6->Checked = false;
Edit_Baut6->Text = myList->Strings[26];
if (myList->Strings[27] == 1) CheckBox7->Checked = true;
else CheckBox7->Checked = false;
Edit_Baut7->Text = myList->Strings[28];
if (myList->Strings[29] == 1) CheckBox8->Checked = true;
else CheckBox8->Checked = false;
Edit_Baut8->Text = myList->Strings[30];
if (myList->Strings[31] == 1) CheckBox9->Checked = true;
else CheckBox9->Checked = false;
Edit_Baut9->Text = myList->Strings[32];
if (myList->Strings[33] == 1) CheckBox10->Checked = true;
else CheckBox10->Checked = false;
Edit_Baut10->Text = myList->Strings[34];

p->StepIt();
int iii = 38; //Start StringGrid in Datei
for (int ii = 1; ii <= 10; ii++)
 {
 for (int i = 1; i <= AnzahlZeilen; i++)
  {
   for (int j = 1; j <= 11; j++)
   {
    switch (ii)
    {
    case  1 : pStringGrid_Baut1->Cells[j][i] = myList->Strings[iii];
              break;
    case  2 : pStringGrid_Baut2->Cells[j][i] = myList->Strings[iii];
              break;
    case  3 : pStringGrid_Baut3->Cells[j][i] = myList->Strings[iii];
              break;
    case  4 : pStringGrid_Baut4->Cells[j][i] = myList->Strings[iii];
              break;
    case  5 : pStringGrid_Baut5->Cells[j][i] = myList->Strings[iii];
              break;
    case  6 : pStringGrid_Baut6->Cells[j][i] = myList->Strings[iii];
```

```
            break;
    case  7 : pStringGrid_Baut7->Cells[j][i] = myList->Strings[iii];
            break;
    case  8 : pStringGrid_Baut8->Cells[j][i] = myList->Strings[iii];
            break;
    case  9 : pStringGrid_Baut9->Cells[j][i] = myList->Strings[iii];
            break;
    case  10: pStringGrid_Baut10->Cells[j][i] = myList->Strings[iii];
            break;
    };
    iii++;
   }
   switch (ii)
   {
    case  1 : CheckForDevice(i,pStringGrid_Baut1);
            break;
    case  2 : CheckForDevice(i,pStringGrid_Baut2);
            break;
    case  3 : CheckForDevice(i,pStringGrid_Baut3);
            break;
    case  4 : CheckForDevice(i,pStringGrid_Baut4);
            break;
    case  5 : CheckForDevice(i,pStringGrid_Baut5);
            break;
    case  6 : CheckForDevice(i,pStringGrid_Baut6);
            break;
    case  7 : CheckForDevice(i,pStringGrid_Baut7);
            break;
    case  8 : CheckForDevice(i,pStringGrid_Baut8);
            break;
    case  9 : CheckForDevice(i,pStringGrid_Baut9);
            break;
    case  10: CheckForDevice(i,pStringGrid_Baut10);
            break;
    };
   }
  p->StepIt();
  }

Edit_SW_VK->Text = myList->Strings[iii];
iii++;

for (int i = 1 ; i <=StringGrid_VK->RowCount; i++)
    for (int j = 0 ; j <=StringGrid_VK->ColCount; j++)
        StringGrid_VK->Cells[j][i] = "";

   int h,g;
   g = (MyListCount - iii)/3;
   StringGrid_VK->RowCount = g+1;
   for (h=0; h < (g); h++)
    {
     StringGrid_VK->Cells[0][h+1] = myList->Strings[iii+0+h*3];
     StringGrid_VK->Cells[1][h+1] = myList->Strings[iii+1+h*3];
     StringGrid_VK->Cells[2][h+1] = myList->Strings[iii+2+h*3];
    }

  p->Position = 0;
  delete myList;
  delete p;
}
```

```
//---------------------------------------------------------------------------
// **************Hauptprogramm**************
//---------------------------------------------------------------------------
__fastcall TF_Main::TF_Main(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TF_Main::Exit1Click(TObject *Sender)
{
exit(0);
}
//---------------------------------------------------------------------------

void __fastcall TF_Main::TreeView1DblClick(TObject *Sender)
{

//Projekt öffnen
if (TreeView1->Items->Item[1]->Selected)
{
OpenDialog1->Filter="Projekt-Datein *.prj|*.prj";
//OpenDialog1->Execute();    Wird in der If Abfrage aufgerufen
if (OpenDialog1->Execute())
  // Memo1->Lines->Add(OpenDialog1->FileName); Dateinamen einfügen
   // Memo1->Lines->LoadFromFile(OpenDialog1->FileName);
   {   //Reset der eingegebenen Bauteile
       Reset1Click(Sender);
       StatusBar1->Panels->Items[1]->Text = OpenDialog1->FileName;
       F_Main->Caption = "DI_Arbeit   " + OpenDialog1->FileName;
       Lesen(OpenDialog1->FileName);
       // Nach öffnen in der Eingabemaske auf default Bauteil1 und laden von Bauteil1 in StringGrid1
       RadioButtonBautSav = -1;
       RadioGroup1->ItemIndex = 0;
       AenderungDurchgefuhert = false;
       //eingegebenen Bauteile übernehmen ? Sollte noch eine abfrage kommen
      if (OKBottomDlg->ShowModal() == mrOk) Button7Click(Sender);
   }
}

//Projekt anlegen
if (TreeView1->Items->Item[2]->Selected)
{
  PageControl1->Visible = true;

  Panel4->Visible = false;

  Panel5->Visible = false;

  Panel1->Visible = false;
  pComboEditor->Visible = false;

  Panel3->Visible = false;

  Panel2->Visible = false;
  Table1->Active = false;

  Panel6->Visible = false;
  Table2->Active = false;
```

```
  OptionenPanel->Visible = false;

}

//Projekt speichern
if (TreeView1->Items->Item[3]->Selected)
{
SaveDialog1->Filter="Projekt-Datein *.prjl*.prj";
if (SaveDialog1->Execute())
   {
   //Memo1->Lines->SaveToFile(SaveDialog1->FileName);
      Schreiben(SaveDialog1->FileName);
      StatusBar1->Panels->Items[1]->Text = SaveDialog1->FileName;
      F_Main->Caption = "DI_Arbeit   " + SaveDialog1->FileName;
      AenderungDurchgefuhert = false;
   }
}

//Adressverwaltung
if (TreeView1->Items->Item[4]->Selected)
{
  PageControl1->Visible = false;

  Panel4->Visible = true;

  Panel5->Visible = false;

  Panel1->Visible = false;
  pComboEditor->Visible = false;

  Panel3->Visible = false;

  Panel2->Visible = false;
  Table1->Active = false;

  Panel6->Visible = false;
  Table2->Active = false;

  OptionenPanel->Visible = false;

  AufrufAdressver();
}

//Programm Optionen
if (TreeView1->Items->Item[7]->Selected)
{
  PageControl1->Visible = false;

  Panel4->Visible = false;

  Panel5->Visible = false;

  Panel1->Visible = false;
  pComboEditor->Visible = false;

  Panel3->Visible = false;

  Panel2->Visible = false;
  Table1->Active = false;
```

```
  Panel6->Visible = false;
  Table2->Active = false;

  OptionenPanel->Visible = true;

  ComboBox_DDC_Option->Clear();
  Table1->Open();
  Query1->Close();
  Query1->SQL->Clear();
  AnsiString ASSQLCommand = "SELECT * FROM Angebot_DB2_Aktuell WHERE Bezeichnung LIKE
'%CPU%' ORDER BY ID";
  Query1->SQL->Add(ASSQLCommand);
  Query1->Open();
  Query1->Active = true;
  Query1->First();
  //ComboBox_DDC_Option befüllen
  while (!Query1->Eof)
   {
    ComboBox_DDC_Option->Items->Add(Query1->FieldByName("Bezeichnung")->AsString);
    Query1->Next();
   }

}

//Programm Ende
if (TreeView1->Items->Item[8]->Selected)
{
exit(0);
}

//LV_Eingeben/bearbeiten
if (TreeView1->Items->Item[10]->Selected)
 {
  PageControl1->Visible = false;

  Panel4->Visible = false;

  Panel5->Visible = false;

  Panel1->Visible = true;

  Panel3->Visible = false;

  Panel2->Visible = false;
  Table1->Active = false;

  Panel6->Visible = false;
  Table2->Active = false;

  OptionenPanel->Visible = false;
 }

//Übersicht
if (TreeView1->Items->Item[14]->Selected)
 {
  PageControl1->Visible = false;

  Panel4->Visible = false;

  Panel5->Visible = false;
```

```
  Panel1->Visible = false;
  pComboEditor->Visible = false;

  Panel3->Visible = true;

  Panel2->Visible = false;
  Table1->Active = false;

  Panel6->Visible = false;
  Table2->Active = false;

  OptionenPanel->Visible = false;
 }

//Rabattblatt
if (TreeView1->Items->Item[15]->Selected)
{
  PageControl1->Visible = false;

  Panel4->Visible = false;

  Panel5->Visible = true;

  Panel1->Visible = false;
  pComboEditor->Visible = false;

  Panel3->Visible = false;

  Panel2->Visible = false;
  Table1->Active = false;

  Panel6->Visible = false;
  Table2->Active = false;

  OptionenPanel->Visible = false;

  AufrufRabattblatt();
}

//DB_Elementenstamm bearbeiten
if (TreeView1->Items->Item[21]->Selected)
 {
  PageControl1->Visible = false;

  Panel4->Visible = false;

  Panel5->Visible = false;

  Panel1->Visible = false;
  pComboEditor->Visible = false;

  Panel3->Visible = false;

  Panel2->Visible = true;
  Table1->Active = true;

  Panel6->Visible = false;
  Table2->Active = false;
```

```
  OptionenPanel->Visible = false;
 }
//DB_2 Elementenstamm bearbeiten
if (TreeView1->Items->Item[22]->Selected)
 {
  PageControl1->Visible = false;

  Panel4->Visible = false;

  Panel5->Visible = false;

  Panel1->Visible = false;
  pComboEditor->Visible = false;

  Panel3->Visible = false;

  Panel2->Visible = false;
  Table1->Active = false;

  Panel6->Visible = true;
  Table2->Active = true;

  OptionenPanel->Visible = false;
 }

}
//---------------------------------------------------------------------------
void __fastcall TF_Main::TreeView1Click(TObject *Sender)
{
//PageControl1->Visible = false;
}
//---------------------------------------------------------------------------

void __fastcall TF_Main::Version1Click(TObject *Sender)
{
//Infobox anzeigen
AboutBox->ShowModal();
}
//---------------------------------------------------------------------------
void __fastcall TF_Main::Deutsch1Click(TObject *Sender)
{
Sprache(true);      //eigene Fkt
}
//---------------------------------------------------------------------------
void __fastcall TF_Main::Englisch1Click(TObject *Sender)
{
Sprache(false);     //eigene Fkt
}
//---------------------------------------------------------------------------

void __fastcall TF_Main::Timer1Timer(TObject *Sender)
{
//Datum und Uhrzeit in der Statusleiste anzeigen
char days[7][11] = {"Sonntag", "Montag", "Dienstag", "Mittwoch", "Donnerstag", "Freitag", "Samstag" };
TDateTime dt=Now();
StatusBar1->Panels->Items[2]->Text =days[dt.DayOfWeek() - 1] + dt.FormatString(", dd.mm.yyyy   hh:nn:ss");

Label1->Visible = false;

if (AenderungDurchgefuhert && FMainCaption == false)
```

```
 {
  F_Main->Caption = F_Main->Caption + "*";
  FMainCaption = true;
 }
if (AenderungDurchgefuhert == false) FMainCaption = false;

}

//---------------------------------------------------------------------------
// OnClick-Eventhandler der Tabelle
//---------------------------------------------------------------------------
void __fastcall TF_Main::StringGrid1Click(TObject *Sender)
{
 // Sender-Zeiger casten:
 TStringGrid* pGrid = dynamic_cast<TStringGrid*>(Sender);

 // Falls 2. Spalte, direkte Bearbeitung der Zelle erlauben:
//  if(pGrid->Col == 1) pGrid->Options  = pGrid->Options << goRowSelect;
//  else pGrid->Options  = pGrid->Options >> goRowSelect;
 if(pGrid->Col == 1)
  {
  pGrid->Options  = pGrid->Options << goEditing;
  AenderungDurchgefuhert = true;
  }
 else pGrid->Options  = pGrid->Options >> goEditing;
 // Falls Spalte 3. bis 12. Spalte
 if(pGrid->Col >= 2 && pGrid->Col <= 11)
 {
  // die Inplace-Combobox füllen:
  if(pGrid->Col == 2)
          LoadGrobToCombo(pComboEditor);
  else
          LoadFein1ToCombo(pComboEditor,pGrid->Col,pGrid->Row);
  // die Inplace-Combobox in der
  // selektierten Zelle anzeigen:
  PlaceComboEditor();

  // Falls der Zelleninhalt in der ComboBox vorhanden ist,
  // entsprechendes Item in der ComboBox anzeigen:
  if(pComboEditor->Items->IndexOf(pGrid->Cells[pGrid->Col][pGrid->Row]) >= 0)
            pComboEditor->ItemIndex = pComboEditor->Items->
          IndexOf(pGrid->Cells[pGrid->Col][pGrid->Row]);
 }
 else
 {
          pComboEditor->Visible = false;
 }

 SaveToPStrGrid();
}

//---------------------------------------------------------------------------
// Füllen der ComboBox mit Grobauswahl
//---------------------------------------------------------------------------
void __fastcall TF_Main::LoadGrobToCombo(TComboBox *pComboBox)
{
 pComboBox->Clear();
 Table1->Open();
 Query1->Close();
 Query1->SQL->Clear();
```

```
 AnsiString ASSQLCommand = "SELECT Grobauswahl FROM Angebot_DB_Aktuell GROUP BY
Grobauswahl ORDER BY Grobauswahl";
 Query1->SQL->Add(ASSQLCommand);
 Query1->Open();
 Query1->Active = true;
 Query1->First();
 //Grobauswahl in ComboBox eintragen
 while (!Query1->Eof)
 {
   pComboEditor->Items->Add(Query1->FieldByName("Grobauswahl")->AsString);
   Query1->Next();
 }
}

//---------------------------------------------------------------------------
// Füllen der ComboBox mit den Fein1
//---------------------------------------------------------------------------
void __fastcall TF_Main::LoadFein1ToCombo(TComboBox *pComboBox, int ACol, int ARow)
{
 pComboBox->Clear();
 Table1->Open();
 Query1->Close();
 Query1->SQL->Clear();
 AnsiString ASSQLCommand;
 bool abbruch = false;

 switch (ACol)
 {
  case 3: ASSQLCommand = "SELECT Fein1 FROM Angebot_DB_Aktuell"
                  " WHERE Grobauswahl = '" + StringGrid1->Cells[ACol-1][ARow] + "'"
                  " GROUP BY Fein1 ORDER BY Fein1";
        break;

  case 4: ASSQLCommand = "SELECT Fein2 FROM Angebot_DB_Aktuell"
                  " WHERE Grobauswahl = '" + StringGrid1->Cells[ACol-2][ARow] + "'"
                  " AND Fein1 = '" + StringGrid1->Cells[ACol-1][ARow] + "'"
                  " GROUP BY Fein2 ORDER BY Fein2";
        break;

  case 5: ASSQLCommand = "SELECT Fein3 FROM Angebot_DB_Aktuell"
                  " WHERE Grobauswahl = '" + StringGrid1->Cells[ACol-3][ARow] + "'"
                  " AND Fein1 = '" + StringGrid1->Cells[ACol-2][ARow] + "'"
                  " AND Fein2 = '" + StringGrid1->Cells[ACol-1][ARow] + "'"
                  " GROUP BY Fein3 ORDER BY Fein3";
        break;

  case 6: ASSQLCommand = "SELECT Fein4 FROM Angebot_DB_Aktuell"
                  " WHERE Grobauswahl = '" + StringGrid1->Cells[ACol-4][ARow] + "'"
                  " AND Fein1 = '" + StringGrid1->Cells[ACol-3][ARow] + "'"
                  " AND Fein2 = '" + StringGrid1->Cells[ACol-2][ARow] + "'"
                  " AND Fein3 = '" + StringGrid1->Cells[ACol-1][ARow] + "'"
                  " GROUP BY Fein4 ORDER BY Fein4";
        break;

  case 7: ASSQLCommand = "SELECT Fein5 FROM Angebot_DB_Aktuell"
                  " WHERE Grobauswahl = '" + StringGrid1->Cells[ACol-5][ARow] + "'"
                  " AND Fein1 = '" + StringGrid1->Cells[ACol-4][ARow] + "'"
                  " AND Fein2 = '" + StringGrid1->Cells[ACol-3][ARow] + "'"
                  " AND Fein3 = '" + StringGrid1->Cells[ACol-2][ARow] + "'"
                  " AND Fein4 = '" + StringGrid1->Cells[ACol-1][ARow] + "'"
```

```
                " GROUP BY Fein5 ORDER BY Fein5";
        break;

 case 8: ASSQLCommand = "SELECT Fein6 FROM Angebot_DB_Aktuell"
                " WHERE Grobauswahl = '" + StringGrid1->Cells[ACol-6][ARow] + "'"
                " AND Fein1 = '" + StringGrid1->Cells[ACol-5][ARow] + "'"
                " AND Fein2 = '" + StringGrid1->Cells[ACol-4][ARow] + "'"
                " AND Fein3 = '" + StringGrid1->Cells[ACol-3][ARow] + "'"
                " AND Fein4 = '" + StringGrid1->Cells[ACol-2][ARow] + "'"
                " AND Fein5 = '" + StringGrid1->Cells[ACol-1][ARow] + "'"
                " GROUP BY Fein6 ORDER BY Fein6";
        break;

 case 9: ASSQLCommand = "SELECT Fein7 FROM Angebot_DB_Aktuell"
                " WHERE Grobauswahl = '" + StringGrid1->Cells[ACol-7][ARow] + "'"
                " AND Fein1 = '" + StringGrid1->Cells[ACol-6][ARow] + "'"
                " AND Fein2 = '" + StringGrid1->Cells[ACol-5][ARow] + "'"
                " AND Fein3 = '" + StringGrid1->Cells[ACol-4][ARow] + "'"
                " AND Fein4 = '" + StringGrid1->Cells[ACol-3][ARow] + "'"
                " AND Fein5 = '" + StringGrid1->Cells[ACol-2][ARow] + "'"
                " AND Fein6 = '" + StringGrid1->Cells[ACol-1][ARow] + "'"
                " GROUP BY Fein7 ORDER BY Fein7";
        break;

 case 10: ASSQLCommand ="SELECT Fein8 FROM Angebot_DB_Aktuell"
                " WHERE Grobauswahl = '" + StringGrid1->Cells[ACol-8][ARow] + "'"
                " AND Fein1 = '" + StringGrid1->Cells[ACol-7][ARow] + "'"
                " AND Fein2 = '" + StringGrid1->Cells[ACol-6][ARow] + "'"
                " AND Fein3 = '" + StringGrid1->Cells[ACol-5][ARow] + "'"
                " AND Fein4 = '" + StringGrid1->Cells[ACol-4][ARow] + "'"
                " AND Fein5 = '" + StringGrid1->Cells[ACol-3][ARow] + "'"
                " AND Fein6 = '" + StringGrid1->Cells[ACol-2][ARow] + "'"
                " AND Fein7 = '" + StringGrid1->Cells[ACol-1][ARow] + "'"
                " GROUP BY Fein8 ORDER BY Fein8";
        break;

 case 11: ASSQLCommand ="SELECT Fein9 FROM Angebot_DB_Aktuell"
                " WHERE Grobauswahl = '" + StringGrid1->Cells[ACol-9][ARow] + "'"
                " AND Fein1 = '" + StringGrid1->Cells[ACol-8][ARow] + "'"
                " AND Fein2 = '" + StringGrid1->Cells[ACol-7][ARow] + "'"
                " AND Fein3 = '" + StringGrid1->Cells[ACol-6][ARow] + "'"
                " AND Fein4 = '" + StringGrid1->Cells[ACol-5][ARow] + "'"
                " AND Fein5 = '" + StringGrid1->Cells[ACol-4][ARow] + "'"
                " AND Fein6 = '" + StringGrid1->Cells[ACol-3][ARow] + "'"
                " AND Fein7 = '" + StringGrid1->Cells[ACol-2][ARow] + "'"
                " AND Fein8 = '" + StringGrid1->Cells[ACol-1][ARow] + "'"
                " GROUP BY Fein9 ORDER BY Fein9";
        break;

 default: abbruch = true; return;
}
Query1->SQL->Add(ASSQLCommand);
Query1->Open();
Query1->Active = true;
Query1->First();
if (abbruch==false)
 {
  //Auswahl in ComboBox eintragen
  while (!Query1->Eof)
  {
```

```
      pComboEditor->Items->Add(Query1->Fields->Fields[0]->AsString);
     Query1->Next();
    }
   }
}

//---------------------------------------------------------------------------
// Beim Scrollen der Tabelle Steuerelemnte ggf. neu positionieren
//---------------------------------------------------------------------------
void __fastcall TF_Main::StringGrid1TopLeftChanged(TObject *Sender)
{
          StringGrid1->Invalidate();
  if(StringGrid1->Col >= 2 && StringGrid1->Col <= 11)
          PlaceComboEditor();
}


//---------------------------------------------------------------------------
// Positionierung der ComboBox in der ausgewählten Zelle
//---------------------------------------------------------------------------
void __fastcall TF_Main::PlaceComboEditor(void)
{
  TRect Rect = StringGrid1->CellRect(StringGrid1->Col,StringGrid1->Row);
  pComboEditor->Top = StringGrid1->Top + Panel1->Top;
  pComboEditor->Left = StringGrid1->Left + Panel1->Left;
  pComboEditor->Text = EmptyStr;
  pComboEditor->Top = pComboEditor->Top + Rect.Top +
    StringGrid1->GridLineWidth;
  pComboEditor->Left = pComboEditor->Left +
    Rect.Left + StringGrid1->GridLineWidth+1;
  pComboEditor->Height = (Rect.Bottom - Rect.Top) + 1;
  pComboEditor->Width = (Rect.Right - Rect.Left) + 0;
  pComboEditor->Visible = true;
 // Setzt Focus von ComboBox - noch schaun ob sinnvoll!!!????
 // pComboEditor->SetFocus();
}

//---------------------------------------------------------------------------
// Eventhandler für die Änderung der Auswahl in der Inplace-ComboBox
//---------------------------------------------------------------------------
void __fastcall TF_Main::ComboEditorChange(TObject *Sender)
{
  // den in der ComboBox selektirten String in die Zelle übernehmen:
  StringGrid1->Cells[StringGrid1->Col][StringGrid1->Row] =
          pComboEditor->Items->Strings[pComboEditor->ItemIndex];
  // löschen der Felder rechts davon
  for (int i = StringGrid1->Col+1; i <=26; i++)
    StringGrid1->Cells[i][StringGrid1->Row] = "";
  //löschen von OK
  StringGrid1->Cells[0][StringGrid1->Row] = "";
  CheckForDevice(StringGrid1->Row,StringGrid1);
 // SaveToPStrGrid();
  StringGrid1->Repaint();
  AenderungDurchgefuhert = true;
}
//---------------------------------------------------------------------------
// Speichern in verdeckte StringGrids nach Änderung
//---------------------------------------------------------------------------
void __fastcall TF_Main::SaveToPStrGrid(void)
{
```

```
int iRow, iCol;
//Speichern der Information in eines der 10 verdeckten StringGrids
switch (RadioGroup1->ItemIndex)
{
 case 0 :
  // Bauteil 1 abgewählt
  for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
   {
   for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
    {
     pStringGrid_Baut1->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
    }
   }
  break;
 case 1 :
  // Bauteil 2 abgewählt
  for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
   {
   for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
    {
     pStringGrid_Baut2->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
    }
   }
  break;
 case 2 :
  // Bauteil 3 abgewählt
  for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
   {
   for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
    {
     pStringGrid_Baut3->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
    }
   }
  break;
 case 3 :
  // Bauteil 4 abgewählt
  for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
   {
   for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
    {
     pStringGrid_Baut4->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
    }
   }
  break;
 case 4 :
  // Bauteil 5 abgewählt
  for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
   {
   for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
    {
     pStringGrid_Baut5->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
    }
   }
  break;
 case 5 :
  // Bauteil 6 abgewählt
  for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
   {
   for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
    {
```

```
      pStringGrid_Baut6->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
     }
    }
   break;
  case 6 :
   // Bauteil 7 abgewählt
   for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
    {
    for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
     {
      pStringGrid_Baut7->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
     }
    }
   break;
  case 7 :
   // Bauteil 8 abgewählt
   for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
    {
    for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
     {
      pStringGrid_Baut8->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
     }
    }
   break;
  case 8 :
   // Bauteil 9 abgewählt
   for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
    {
    for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
     {
      pStringGrid_Baut9->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
     }
    }
   break;
  case 9 :
   // Bauteil 10 abgewählt
   for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
    {
    for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
     {
      pStringGrid_Baut10->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
     }
    }
   break;
}
}
//---------------------------------------------------------------------------
// Überprüfen auf vollständige Zeile
//---------------------------------------------------------------------------
void __fastcall TF_Main::CheckForDevice(int ARow, TStringGrid *StrGd)
{
 StrGd->Cells[0][ARow] = "";
 int i=1;
 AnsiString ASSQLCommand ;
 do
 {
  Table1->Open();
  Query1->Close();
  Query1->SQL->Clear();
  switch (i)
```

```
{
 case 1: ASSQLCommand = "SELECT * FROM Angebot_DB_Aktuell"
                  " WHERE Grobauswahl = '" + StrGd->Cells[2][ARow] + "'";
        break;

 case 2: if (StrGd->Cells[3][ARow] == "")
           {
           i=-1;
           break;
           };
        ASSQLCommand = "SELECT * FROM Angebot_DB_Aktuell"
                  " WHERE Grobauswahl = '" + StrGd->Cells[2][ARow] +
                  "' AND Fein1 = '" + StrGd->Cells[3][ARow] + "' ";
        break;

 case 3: if (StrGd->Cells[4][ARow] == "")
           {
           i=-1;
           break;
           };
        ASSQLCommand = "SELECT * FROM Angebot_DB_Aktuell"
                  " WHERE Grobauswahl = '" + StrGd->Cells[2][ARow] +
                  "' AND Fein1 = '" + StrGd->Cells[3][ARow] +
                  "' AND Fein2 = '" + StrGd->Cells[4][ARow] + "' ";
        break;

 case 4: if (StrGd->Cells[5][ARow] == "")
           {
           i=-1;
           break;
           };
        ASSQLCommand = "SELECT * FROM Angebot_DB_Aktuell"
                  " WHERE Grobauswahl = '" + StrGd->Cells[2][ARow] +
                  "' AND Fein1 = '" + StrGd->Cells[3][ARow] +
                  "' AND Fein2 = '" + StrGd->Cells[4][ARow] +
                  "' AND Fein3 = '" + StrGd->Cells[5][ARow] + "' ";
        break;

 case 5: if (StrGd->Cells[6][ARow] == "")
           {
           i=-1;
           break;
           };
        ASSQLCommand = "SELECT * FROM Angebot_DB_Aktuell"
                  " WHERE Grobauswahl = '" + StrGd->Cells[2][ARow] +
                  "' AND Fein1 = '" + StrGd->Cells[3][ARow] +
                  "' AND Fein2 = '" + StrGd->Cells[4][ARow] +
                  "' AND Fein3 = '" + StrGd->Cells[5][ARow] +
                  "' AND Fein4 = '" + StrGd->Cells[6][ARow] + "' ";
        break;

 case 6: if (StrGd->Cells[7][ARow] == "")
           {
           i=-1;
           break;
           };
        ASSQLCommand = "SELECT * FROM Angebot_DB_Aktuell"
                  " WHERE Grobauswahl = '" + StrGd->Cells[2][ARow] +
                  "' AND Fein1 = '" + StrGd->Cells[3][ARow] +
                  "' AND Fein2 = '" + StrGd->Cells[4][ARow] +
```

```
                "' AND Fein3 = '" + StrGd->Cells[5][ARow] +
                "' AND Fein4 = '" + StrGd->Cells[6][ARow] +
                "' AND Fein5 = '" + StrGd->Cells[7][ARow] + "' ";
        break;

case 7: if (StrGd->Cells[8][ARow] == "")
          {
          i=-1;
          break;
          };
        ASSQLCommand = "SELECT * FROM Angebot_DB_Aktuell"
                " WHERE Grobauswahl = '" + StrGd->Cells[2][ARow] +
                "' AND Fein1 = '" + StrGd->Cells[3][ARow] +
                "' AND Fein2 = '" + StrGd->Cells[4][ARow] +
                "' AND Fein3 = '" + StrGd->Cells[5][ARow] +
                "' AND Fein4 = '" + StrGd->Cells[6][ARow] +
                "' AND Fein5 = '" + StrGd->Cells[7][ARow] +
                "' AND Fein6 = '" + StrGd->Cells[8][ARow] + "' ";
        break;

case 8: if (StrGd->Cells[9][ARow] == "")
          {
          i=-1;
          break;
          };
        ASSQLCommand = "SELECT * FROM Angebot_DB_Aktuell"
                " WHERE Grobauswahl = '" + StrGd->Cells[2][ARow] +
                "' AND Fein1 = '" + StrGd->Cells[3][ARow] +
                "' AND Fein2 = '" + StrGd->Cells[4][ARow] +
                "' AND Fein3 = '" + StrGd->Cells[5][ARow] +
                "' AND Fein4 = '" + StrGd->Cells[6][ARow] +
                "' AND Fein5 = '" + StrGd->Cells[7][ARow] +
                "' AND Fein6 = '" + StrGd->Cells[8][ARow] +
                "' AND Fein7 = '" + StrGd->Cells[9][ARow] + "' ";
        break;

case 9: if (StrGd->Cells[10][ARow] == "")
          {
          i=-1;
          break;
          };
        ASSQLCommand = "SELECT * FROM Angebot_DB_Aktuell"
                " WHERE Grobauswahl = '" + StrGd->Cells[2][ARow] +
                "' AND Fein1 = '" + StrGd->Cells[3][ARow] +
                "' AND Fein2 = '" + StrGd->Cells[4][ARow] +
                "' AND Fein3 = '" + StrGd->Cells[5][ARow] +
                "' AND Fein4 = '" + StrGd->Cells[6][ARow] +
                "' AND Fein5 = '" + StrGd->Cells[7][ARow] +
                "' AND Fein6 = '" + StrGd->Cells[8][ARow] +
                "' AND Fein7 = '" + StrGd->Cells[9][ARow] +
                "' AND Fein8 = '" + StrGd->Cells[10][ARow] + "' ";
        break;

case 10: ASSQLCommand = "SELECT * FROM Angebot_DB_Aktuell"
                " WHERE Grobauswahl = '" + StrGd->Cells[2][ARow] +
                "' AND Fein1 = '" + StrGd->Cells[3][ARow] +
                "' AND Fein2 = '" + StrGd->Cells[4][ARow] +
                "' AND Fein3 = '" + StrGd->Cells[5][ARow] +
                "' AND Fein4 = '" + StrGd->Cells[6][ARow] +
                "' AND Fein5 = '" + StrGd->Cells[7][ARow] +
```

```
                    "' AND Fein6 = '" + StrGd->Cells[8][ARow] +
                    "' AND Fein7 = '" + StrGd->Cells[9][ARow] +
                    "' AND Fein8 = '" + StrGd->Cells[10][ARow] +
                    "' AND Fein9 = '" + StrGd->Cells[11][ARow] + "' ";
          break;

   default: i =-1; return;
  }
  Query1->SQL->Add(ASSQLCommand);
  Query1->Open();
  Query1->Active = true;
  Query1->First();
  i++;
 } while(Query1->RecordCount != 1 && i != 0);
 if (Query1->RecordCount == 1)
     {
     StrGd->Cells[12][ARow] = Query1->FieldByName("Gerätetyp")->AsString;
     StrGd->Cells[13][ARow] = Query1->FieldByName("Gerätetyp2")->AsString;
     StrGd->Cells[14][ARow] = Query1->FieldByName("Gerätetyp3")->AsString;
     StrGd->Cells[15][ARow] = Query1->FieldByName("AE")->AsInteger;
     StrGd->Cells[16][ARow] = Query1->FieldByName("AA")->AsInteger;
     StrGd->Cells[17][ARow] = Query1->FieldByName("DE")->AsInteger;
     StrGd->Cells[18][ARow] = Query1->FieldByName("DA")->AsInteger;
     StrGd->Cells[19][ARow] = Query1->FieldByName("Hersteller")->AsString;
     StrGd->Cells[20][ARow] = Query1->FieldByName("RabattgruppeEK")->AsString;
     StrGd->Cells[21][ARow] = Query1->FieldByName("RabattgruppeVK")->AsString;
     StrGd->Cells[22][ARow] = Query1->FieldByName("ListenpreisType1")->AsCurrency;
     StrGd->Cells[23][ARow] = Query1->FieldByName("ListenpreisType2")->AsCurrency;
     StrGd->Cells[24][ARow] = Query1->FieldByName("ListenpreisType3")->AsCurrency;
     StrGd->Cells[25][ARow] = Query1->FieldByName("BMGruppe")->AsString;
     StrGd->Cells[26][ARow] = Query1->FieldByName("Ausschreibungstext")->AsString;
     StrGd->Cells[0][ARow] = "OK";
     Label1->Visible = true;
     };

}
//---------------------------------------------------------------------------
// OnResize-Eventhandler des Formulars
//---------------------------------------------------------------------------
void __fastcall TF_Main::FormResize(TObject *Sender)
{
  // Breite der letzten Spalte an die Formularbreite anpassen:  ???
          int ilColWidthsSum = 0;
  for(int ilCol = 0; ilCol < StringGrid1->ColCount; ilCol++)
    ilColWidthsSum+=StringGrid1->ColWidths[ilCol]+StringGrid1->GridLineWidth;

  if(ilColWidthsSum < StringGrid1->ClientWidth)
          StringGrid1->ColWidths[StringGrid1->ColCount-1] =
          StringGrid1->ClientWidth - ilColWidthsSum +
          StringGrid1->ColWidths[StringGrid1->ColCount-1];

Panel3->Width=(F_Main->Width - Panel3->Left - 25);
PageControl2->Width=(Panel3->Width - PageControl2->Left - 25);
RichEdit_Gesamt->Width=(PageControl2->Width - RichEdit_Gesamt->Left - 25);
RichEdit_Kalkulationsblatt->Width=(PageControl2->Width - RichEdit_Kalkulationsblatt->Left - 25);

}
//---------------------------------------------------------------------------
// HintergrundFarbe der Zeilen in StringGrid definieren
//---------------------------------------------------------------------------
```

```
void __fastcall TF_Main::StringGrid1DrawCell(TObject *Sender, int ACol,
     int ARow, TRect &Rect, TGridDrawState State)
{
 // Farben festlegen:
 // Standardfarben für selektierte Zelle(n)
 if (State.Contains(gdSelected))
 {
  StringGrid1->Canvas->Brush->Color = clHighlight;
  StringGrid1->Canvas->Font->Color = clHighlightText;
 }
 // gerade Zeilen Dunkelgrau, aber nicht die feste(n) Zeile(n) oben
 else if (StringGrid1->Cells[0][ARow] == "OK" && !State.Contains(gdFixed))
 {
  StringGrid1->Canvas->Brush->Color = clNavy;//clLtGray
  StringGrid1->Canvas->Font->Color = clWhite;
 }

 // Zelle endlich zeichnen
 StringGrid1->Canvas->FillRect(Rect);

 // noch ein Gimmick für die Textausrichtung
 int hAlign = 1;   // text align, 0/1/2 - Left/Center/Right

 // und zum Schluss den Text in die Zelle malen
 DrawText(StringGrid1->Canvas->Handle, StringGrid1->Cells[ACol][ARow].c_str(),
   -1, &Rect, DT_SINGLELINE | DT_VCENTER | hAlign);

}
//---------------------------------------------------------------------------
// DB in ExcelListe exportieren
//---------------------------------------------------------------------------

void __fastcall TF_Main::Button1Click(TObject *Sender)
{

Table1->Active = false;

Variant Excel;

try
 {
 Excel = GetActiveOleObject("Excel.Application");
 }
catch(...)
 {
 Excel = CreateOleObject("Excel.Application");
 }
//  Excel.OlePropertySet("Visible", true);

try
 {
 Variant WorkBooks = Excel.OlePropertyGet("WorkBooks");
 //WorkBooks.OleFunction("Add");
 WorkBooks.OleFunction("Open",Dir + "\\files\\Exp_Datenbank_orig.xls");
 }
catch(...)
 {
 }
```

```
Variant ActiveWorkBook = Excel.OlePropertyGet("ActiveWorkbook");

Variant WorkSheets = Excel.OlePropertyGet("Worksheets");

Variant WorkSheet = WorkSheets.OlePropertyGet("Item", 1);
WorkSheet.OleFunction("Activate");

Table1->Open();
Query1->Close();
Query1->SQL->Clear();
AnsiString ASSQLCommand = "SELECT * FROM Angebot_DB_Aktuell ORDER BY ID";
Query1->SQL->Add(ASSQLCommand);
Query1->Open();
Query1->Active = true;
Query1->First();

//MaxWert für Fortschrittsanzeige setzen
ProgressBar1->Max = Query1->RecordCount;

int iRow, iCol;
try
 {
  //Beschreiben der Excelliste
  for (iRow=2; iRow <= AnzahlZeilenExpImp; iRow++)
   {
    //Fortschrittsanzeige +1 setzen
    ProgressBar1->StepIt();
    for (iCol=1; iCol <=26; iCol++)
      {
       Variant Range = WorkSheet.OlePropertyGet("Cells", iRow, iCol);
       Range.OlePropertySet("Value", Query1->Fields->Fields[iCol-1]->AsString);
      }
    Query1->Next();
    //Bei Eof Abbruch des Exports
    if (Query1->Eof) break;

   }
  //ShowMessage("Daten wurden in Excel exportiert!");
 }
catch(...)
 {
  ShowMessage("Fehler bei Datenexport! in Zeile: "+IntToStr(iRow)+" Spalte: "+ IntToStr(iCol));
 }

try
 {
 ActiveWorkBook.OleFunction("SaveAs", Dir + "\\Exp_Datenbank.xls");
 }
catch(...)
 {
 }
Excel.OleFunction("Quit");
Excel = Unassigned;

Table1->Active = true;

//Fortschrittsanzeige auf 0 setzen
ProgressBar1->Position = 0;
}
//---------------------------------------------------------------------------
```

```
// ExcelListe in DB importieren
//---------------------------------------------------------------------------

void __fastcall TF_Main::Button2Click(TObject *Sender)
{

Table1->Active = false;

Variant Excel;

try
 {
  Excel = GetActiveOleObject("Excel.Application");
 }
catch(...)
 {
  Excel = CreateOleObject("Excel.Application");
 }
//  Excel.OlePropertySet("Visible", true);

try
 {
  Variant WorkBooks = Excel.OlePropertyGet("WorkBooks");
  //WorkBooks.OleFunction("Add");
  WorkBooks.OleFunction("Open",Dir + "\\Exp_Datenbank.xls");
 }
catch(...)
 {
 }

Variant ActiveWorkBook = Excel.OlePropertyGet("ActiveWorkbook");

Variant WorkSheets = Excel.OlePropertyGet("Worksheets");

Variant WorkSheet = WorkSheets.OlePropertyGet("Item", 1);
WorkSheet.OleFunction("Activate");

int iRow, iCol;
int ptr = 0, ptr1 = 0;
try
 {
  //Auslesen der Excelliste - bis Grenze wg Statusanzeige
  for (iRow=2; iRow <= AnzahlZeilenExpImp; iRow++)
   {
    //Überprüfen letzen Eintrag in Excel
    ptr = AnsiCompareStr (WorkSheet.OlePropertyGet("Cells",iRow, 1).OlePropertyGet("Value"),"");
    if ( ptr == 0)
       break;
   }
  //MaxWert für Fortschrittsanzeige setzen
  ProgressBar1->Max = iRow
  ;
 //Auslesen der Excelliste - Einlesen in die Datenbank
 for (iRow=2; iRow <= AnzahlZeilenExpImp; iRow++)
  {
   //Fortschrittsanzeige +1 setzen
   ProgressBar1->StepIt();
   //Überprüfen letzen Eintrag in Excel
   ptr = AnsiCompareStr (WorkSheet.OlePropertyGet("Cells",iRow, 1).OlePropertyGet("Value"),"");
   if ( ptr == 0)
```

```
      break;
    //ist eine ID nicht vorhanden - neuen Datensatz mit dieser ID erstellen
    Table1->Open();
    Query1->Close();
    Query1->SQL->Clear();
    AnsiString ASSQLCommand = "SELECT * FROM Angebot_DB_Aktuell ORDER BY ID";
    Query1->SQL->Add(ASSQLCommand);
    Query1->Open();
    Query1->Active = true;
    Query1->First();
    while (!Query1->Eof)
     {
      ptr1 = AnsiCompareStr (WorkSheet.OlePropertyGet("Cells",iRow, 1).OlePropertyGet("Value"),Query1->FieldByName("ID")->AsString);
      if (ptr1 == 0)
        break;
      Query1->Next();
     }
    if (ptr1 != 0)
      {
      Table1->Open();
      Query1->Close();
      Query1->SQL->Clear();      //SQL Commando insert Angebot_DB_Aktuell mit der Spalte "ID" mit Wert aus der "aktiven" Celle  wo Id der Zeile
      AnsiString ASSQLCommand = "Insert Into Angebot_DB_Aktuell (ID) VALUES ("+ WorkSheet.OlePropertyGet("Cells",iRow, 1).OlePropertyGet("Value") +")";
      Query1->SQL->Add(ASSQLCommand);
      Query1->ExecSQL();
      }
    //Daten nun Importieren
    for (iCol=1; iCol <=26; iCol++)
     {

      AnsiString AStemp = "";
      switch (iCol)   //In welcher Spalte man sich grad befindet
      {                    //Zuweisung der Spaltenbezeichnung
       case 1: AStemp ="ID";
            break;
       case 2: AStemp ="Grobauswahl";
            break;
       case 3: AStemp ="Fein1";
            break;
       case 4: AStemp ="Fein2";
            break;
       case 5: AStemp ="Fein3";
            break;
       case 6: AStemp ="Fein4";
            break;
       case 7: AStemp ="Fein5";
            break;
       case 8: AStemp ="Fein6";
            break;
       case 9: AStemp ="Fein7";
            break;
       case 10: AStemp ="Fein8";
            break;
       case 11: AStemp ="Fein9";
            break;
       case 12: AStemp ="Gerätetyp";
            break;
```

```
        case 13: AStemp ="Gerätetyp2";
                break;
        case 14: AStemp ="Gerätetyp3";
                break;
        case 15: AStemp ="AE";
                break;
        case 16: AStemp ="AA";
                break;
        case 17: AStemp ="DE";
                break;
        case 18: AStemp ="DA";
                break;
        case 19: AStemp ="Hersteller";
                break;
        case 20: AStemp ="RabattgruppeEK";
                break;
        case 21: AStemp ="RabattgruppeVK";
                break;
        case 22: AStemp ="ListenpreisType1";
                break;
        case 23: AStemp ="ListenpreisType2";
                break;
        case 24: AStemp ="ListenpreisType3";
                break;
        case 25: AStemp ="BMGruppe";
                break;
        case 26: AStemp ="Ausschreibungstext";
                break;
        default: return;
        }
       AnsiString Wert = WorkSheet.OlePropertyGet("Cells",iRow, iCol).OlePropertyGet("Value");
       int i = 0;
       i = Wert.Pos(',');
       // In der Preisspalte muss das Komma durch einen Punkt beim Import ersetzt werden
       if (i != 0 && (iCol == 22 || iCol == 23 || iCol == 24))
         {
          Wert.Delete(i,1);
          Wert.Insert('.',i);
         }
       Table1->Open();
       Query1->Close();
       Query1->SQL->Clear();       //SQL Commando update Angebot_DB_Aktuell mit der Spalte "AStemp" mit
Wert aus der "aktiven" Celle  wo Id der Zeile suchen nach Komma und durch punkt ersetzen: AnsiString.pos
while AnsiString.pos"/delete/insert," durch "."
       AnsiString ASSQLCommand = "Update Angebot_DB_Aktuell SET " + AStemp + "='" + Wert + "'
WHERE ID = '" + WorkSheet.OlePropertyGet("Cells",iRow, 1).OlePropertyGet("Value") +"'";
       Query1->SQL->Add(ASSQLCommand);
       Query1->ExecSQL();
      }
    }
  //ShowMessage("Daten wurden in Excel exportiert!");
 }
catch(...)
 {
  ShowMessage("Fehler bei Datenimport! in Zeile: "+IntToStr(iRow)+" Spalte: "+ IntToStr(iCol));
 }

Excel.OleFunction("Quit");
Excel = Unassigned;
```

```
Table1->Active = true;

//Fortschrittsanzeige auf 0 setzen
ProgressBar1->Position = 0;
}
//---------------------------------------------------------------------------
// Dekonstruktor
//---------------------------------------------------------------------------
__fastcall TF_Main::~TF_Main()
{

SaveDialog1->Filter="Projekt-Datein *.prjl*.prj";
if (SaveDialog1->FileName == "" && AenderungDurchgefuhert)
  {
   if (SaveDialog1->Execute())
     {
      Schreiben(SaveDialog1->FileName);
     }
  }
else if (AenderungDurchgefuhert)
       Schreiben(SaveDialog1->FileName);
delete pPage;
delete pComboEditor;
delete pStringGrid_Baut1;
delete pStringGrid_Baut2;
delete pStringGrid_Baut3;
delete pStringGrid_Baut4;
delete pStringGrid_Baut5;
delete pStringGrid_Baut6;
delete pStringGrid_Baut7;
delete pStringGrid_Baut8;
delete pStringGrid_Baut9;
delete pStringGrid_Baut10;
delete StrGd;

delete pKalkulation_Baut1;
delete pKalkulation_Baut2;
delete pKalkulation_Baut3;
delete pKalkulation_Baut4;
delete pKalkulation_Baut5;
delete pKalkulation_Baut6;
delete pKalkulation_Baut7;
delete pKalkulation_Baut8;
delete pKalkulation_Baut9;
delete pKalkulation_Baut10;

}
//---------------------------------------------------------------------------
// Mauszeige auf StringGrid lokalisieren für PopupMenü
//---------------------------------------------------------------------------

void __fastcall TF_Main::StringGrid1ContextPopup(TObject *Sender,
     TPoint &MousePos, bool &Handled)
{
// eigenes PopupMenü
//PopupMenu1->Popup(Mouse->CursorPos.x,Mouse->CursorPos.y);
//Handled=true;

TPoint globpoint = Mouse->CursorPos; // globale Mauskoordinaten holen
TPoint locpoint = StringGrid1->ScreenToClient(globpoint); // Koordinaten für das Stringgrid umrechnen
```

```
StringGrid1->MouseToCell(locpoint.x, locpoint.y, copycol, copyrow); //Zeilen und Spaltenpositionen holen

}
//---------------------------------------------------------------------------
// PopupMenü aktuelle Zeile kopieren
//---------------------------------------------------------------------------

void __fastcall TF_Main::Kopieren1Click(TObject *Sender)
{

  // aktuelle Zeile kopieren
  Zelle1 = StringGrid1->Cells[1][copyrow];
  Zelle2 = StringGrid1->Cells[2][copyrow];
  Zelle3 = StringGrid1->Cells[3][copyrow];
  Zelle4 = StringGrid1->Cells[4][copyrow];
  Zelle5 = StringGrid1->Cells[5][copyrow];
  Zelle6 = StringGrid1->Cells[6][copyrow];
  Zelle7 = StringGrid1->Cells[7][copyrow];
  Zelle8 = StringGrid1->Cells[8][copyrow];
  Zelle9 = StringGrid1->Cells[9][copyrow];
  Zelle10 = StringGrid1->Cells[10][copyrow];
  Zelle11 = StringGrid1->Cells[11][copyrow];
}
//---------------------------------------------------------------------------
// PopupMenü kopierte Zeile einfügen
//---------------------------------------------------------------------------

void __fastcall TF_Main::Einfgen1Click(TObject *Sender)
{
  // aktuelle Zeile einfügen
  StringGrid1->Cells[1][copyrow] = Zelle1;
  StringGrid1->Cells[2][copyrow] = Zelle2;
  StringGrid1->Cells[3][copyrow] = Zelle3;
  StringGrid1->Cells[4][copyrow] = Zelle4;
  StringGrid1->Cells[5][copyrow] = Zelle5;
  StringGrid1->Cells[6][copyrow] = Zelle6;
  StringGrid1->Cells[7][copyrow] = Zelle7;
  StringGrid1->Cells[8][copyrow] = Zelle8;
  StringGrid1->Cells[9][copyrow] = Zelle9;
  StringGrid1->Cells[10][copyrow] = Zelle10;
  StringGrid1->Cells[11][copyrow] = Zelle11;
for (int i=1; i<=AnzahlZeilen; i++)
  {
   CheckForDevice(i,StringGrid1);
  }
}
//---------------------------------------------------------------------------
// Aufruf Adressverzeichnis
//---------------------------------------------------------------------------

void TF_Main::AufrufAdressver(void)
{
  TListColumn  *NewColumn;
  TListItem  *ListItem;

  //löschen der vorhandenen ListView1
  ListView1->Columns->Clear();
  ListView1->Items->Clear();
  //Einstellungen ListView und Überschriften
  ListView1->ViewStyle = vsReport;
```

```
 ListView1->Top = 100;
 ListView1->Left = 50;
 ListView1->Width = Panel4->Width - 100;
 ListView1->Height = Panel4->Height - 150;
 NewColumn = ListView1->Columns->Add();
 NewColumn->Width = 200;
 NewColumn->Caption = "Firma";
 NewColumn = ListView1->Columns->Add();
 NewColumn->Width = 200;
 NewColumn->Caption = "Name";
 NewColumn = ListView1->Columns->Add();
 NewColumn->Width = 200;
 NewColumn->Caption = "Straße";
 NewColumn = ListView1->Columns->Add();
 NewColumn->Width = 200;
 NewColumn->Caption = "Ort";

 //Laden aus Datei
 TStringList* myList = new TStringList();
 myList->LoadFromFile(Dir+"\\files\\Adress.ver");

 for (int i = 0; i < (myList->Count/4); i++)
  {
   ListItem = ListView1->Items->Add();
   ListItem->Caption = myList->Strings[0+i*4];
   ListItem->SubItems->Add(myList->Strings[1+i*4]);
   ListItem->SubItems->Add(myList->Strings[2+i*4]);
   ListItem->SubItems->Add(myList->Strings[3+i*4]);
  }
delete myList;

}
//---------------------------------------------------------------------------
// selektierte Adresse in Projekt übertragen
//---------------------------------------------------------------------------

void __fastcall TF_Main::Button4Click(TObject *Sender)
{
TListItem  *ListItem;
//Übertragen in Blatt "Projekt anlegen"
if (ListView1->Selected)
 {
  ListItem = ListView1->Selected;
  Edit_Firma->Text = ListItem->Caption;
  Edit_zHd->Text = ListItem->SubItems->Strings[0];
  Edit_Strasse->Text = ListItem->SubItems->Strings[1];
  Edit_Ort->Text = ListItem->SubItems->Strings[2];
  ShowMessage("Adresse wurde ins Blatt 'Projekt anlegen' übertragen");
  AenderungDurchgefuhert = true;
 }
else
 {
  ShowMessage("Keine Adresse ausgewählt");
 }
}
//---------------------------------------------------------------------------
// neue Adresse einfügen/editieren
//---------------------------------------------------------------------------

void __fastcall TF_Main::Button5Click(TObject *Sender)
```

```
{
TListItem  *ListItem;

//Anzeige in Editfeldern
if (ListView1->Selected)
 {
  ListItem = ListView1->Selected;
  EditierenNeuerEintrag->Edit_Firma->Text = ListItem->Caption;
  EditierenNeuerEintrag->Edit_zHd->Text = ListItem->SubItems->Strings[0];
  EditierenNeuerEintrag->Edit_Strasse->Text = ListItem->SubItems->Strings[1];
  EditierenNeuerEintrag->Edit_Ort->Text = ListItem->SubItems->Strings[2];
 }
 else
 {
  EditierenNeuerEintrag->Edit_Firma->Text = "Firma";
  EditierenNeuerEintrag->Edit_zHd->Text = "zHd";
  EditierenNeuerEintrag->Edit_Strasse->Text = "Strasse";
  EditierenNeuerEintrag->Edit_Ort->Text = "Ort";
 }

if (EditierenNeuerEintrag->ShowModal() == mrOk)
 {
  //Editieren eines vorhandenen Eintrags
  if (ListView1->Selected)
   {

    ListItem = ListView1->Selected;
    ListItem->Caption = EditierenNeuerEintrag->Edit_Firma->Text;
    ListItem->SubItems->Strings[0] = EditierenNeuerEintrag->Edit_zHd->Text;
    ListItem->SubItems->Strings[1] = EditierenNeuerEintrag->Edit_Strasse->Text;
    ListItem->SubItems->Strings[2] = EditierenNeuerEintrag->Edit_Ort->Text;
    ShowMessage("Adresse wurde editiert");
   }
  else
   {
    ShowMessage("Keine Adresse ausgewählt");
   }
 }

if (EditierenNeuerEintrag->ModalResult == mrIgnore)
 {
  //Neuer Eintrag zur ListView1
  ListItem = ListView1->Items->Add();
  ListItem->Caption = EditierenNeuerEintrag->Edit_Firma->Text;
  ListItem->SubItems->Add(EditierenNeuerEintrag->Edit_zHd->Text);
  ListItem->SubItems->Add(EditierenNeuerEintrag->Edit_Strasse->Text);
  ListItem->SubItems->Add(EditierenNeuerEintrag->Edit_Ort->Text);
  ShowMessage("Adresse wurde hinzugefügt");
 }

 //Speichern der ListView1 in datei
 TStringList* myList = new TStringList();

  for (int i = 0; i < (ListView1->Items->Count); i++)
   {
    ListItem = ListView1->Items->Item[i];
    myList->Add(ListItem->Caption);
    myList->Add(ListItem->SubItems->Strings[0]);
    myList->Add(ListItem->SubItems->Strings[1]);
    myList->Add(ListItem->SubItems->Strings[2]);
```

```
   }

 myList->SaveToFile(Dir+"\\files\\Adress.ver");
 delete myList;

}
//---------------------------------------------------------------------------
// löschen der selektieren Adresse
//---------------------------------------------------------------------------

void __fastcall TF_Main::Button6Click(TObject *Sender)
{
TListItem  *ListItem;
//Löschen eines selektieren Eintrags aus der ListView1
if (ListView1->Selected)
 {
  ListItem = ListView1->Selected;
  ListItem->Delete();
  ShowMessage("Adresse wurde gelöscht");
 }
else
 {
  ShowMessage("Keine Adresse ausgewählt");
 }

 //Speichern der ListView1 in datei
 TStringList* myList = new TStringList();

  for (int i = 0; i < (ListView1->Items->Count); i++)
   {
    ListItem = ListView1->Items->Item[i];
    myList->Add(ListItem->Caption);
    myList->Add(ListItem->SubItems->Strings[0]);
    myList->Add(ListItem->SubItems->Strings[1]);
    myList->Add(ListItem->SubItems->Strings[2]);
   }

 myList->SaveToFile(Dir+"\\files\\Adress.ver");
 delete myList;
}
//---------------------------------------------------------------------------
// Änderung eines Editfeldes erkennen
//---------------------------------------------------------------------------

void __fastcall TF_Main::Edit_FirmaChange(TObject *Sender)
{
  //Bei Änderung eines EditFeldes immer Speichern verlangen
  AenderungDurchgefuhert = true;
}
//---------------------------------------------------------------------------
// Konfiguration Bauteile CheckBox aktiv - Freigabe Edit u. nächstes Bauteil
//---------------------------------------------------------------------------

void __fastcall TF_Main::CheckBox1Click(TObject *Sender)
{
Edit_Baut1->Enabled=true;
if (CheckBox1->Checked) CheckBox2->Enabled = true;
else
 {
 CheckBox2->Enabled = false;
```

```
 CheckBox2->Checked = false;
 CheckBox3->Enabled = false;
 CheckBox3->Checked = false;
 CheckBox4->Enabled = false;
 CheckBox4->Checked = false;
 CheckBox5->Enabled = false;
 CheckBox5->Checked = false;
 CheckBox6->Enabled = false;
 CheckBox6->Checked = false;
 CheckBox7->Enabled = false;
 CheckBox7->Checked = false;
 CheckBox8->Enabled = false;
 CheckBox8->Checked = false;
 CheckBox9->Enabled = false;
 CheckBox9->Checked = false;
 CheckBox10->Enabled = false;
 CheckBox10->Checked = false;
 }
}
//---------------------------------------------------------------------------

void __fastcall TF_Main::CheckBox2Click(TObject *Sender)
{
Edit_Baut2->Enabled=true;
if (CheckBox2->Checked) CheckBox3->Enabled = true;
else
 {
 CheckBox3->Enabled = false;
 CheckBox3->Checked = false;
 CheckBox4->Enabled = false;
 CheckBox4->Checked = false;
 CheckBox5->Enabled = false;
 CheckBox5->Checked = false;
 CheckBox6->Enabled = false;
 CheckBox6->Checked = false;
 CheckBox7->Enabled = false;
 CheckBox7->Checked = false;
 CheckBox8->Enabled = false;
 CheckBox8->Checked = false;
 CheckBox9->Enabled = false;
 CheckBox9->Checked = false;
 CheckBox10->Enabled = false;
 CheckBox10->Checked = false;
 }
}
//---------------------------------------------------------------------------

void __fastcall TF_Main::CheckBox3Click(TObject *Sender)
{
Edit_Baut3->Enabled=true;
if (CheckBox3->Checked) CheckBox4->Enabled = true;
else
 {
 CheckBox4->Enabled = false;
 CheckBox4->Checked = false;
 CheckBox5->Enabled = false;
 CheckBox5->Checked = false;
 CheckBox6->Enabled = false;
 CheckBox6->Checked = false;
 CheckBox7->Enabled = false;
```

```
 CheckBox7->Checked = false;
 CheckBox8->Enabled = false;
 CheckBox8->Checked = false;
 CheckBox9->Enabled = false;
 CheckBox9->Checked = false;
 CheckBox10->Enabled = false;
 CheckBox10->Checked = false;
 }
}
//---------------------------------------------------------------------------

void __fastcall TF_Main::CheckBox4Click(TObject *Sender)
{
Edit_Baut4->Enabled=true;
if (CheckBox4->Checked) CheckBox5->Enabled = true;
else
 {
 CheckBox5->Enabled = false;
 CheckBox5->Checked = false;
 CheckBox6->Enabled = false;
 CheckBox6->Checked = false;
 CheckBox7->Enabled = false;
 CheckBox7->Checked = false;
 CheckBox8->Enabled = false;
 CheckBox8->Checked = false;
 CheckBox9->Enabled = false;
 CheckBox9->Checked = false;
 CheckBox10->Enabled = false;
 CheckBox10->Checked = false;
 }
}
//---------------------------------------------------------------------------

void __fastcall TF_Main::CheckBox5Click(TObject *Sender)
{
Edit_Baut5->Enabled=true;
if (CheckBox5->Checked) CheckBox6->Enabled = true;
else
 {
 CheckBox6->Enabled = false;
 CheckBox6->Checked = false;
 CheckBox7->Enabled = false;
 CheckBox7->Checked = false;
 CheckBox8->Enabled = false;
 CheckBox8->Checked = false;
 CheckBox9->Enabled = false;
 CheckBox9->Checked = false;
 CheckBox10->Enabled = false;
 CheckBox10->Checked = false;
 }
}
//---------------------------------------------------------------------------

void __fastcall TF_Main::CheckBox6Click(TObject *Sender)
{
Edit_Baut6->Enabled=true;
if (CheckBox6->Checked) CheckBox7->Enabled = true;
else {
 CheckBox7->Enabled = false;
 CheckBox7->Checked = false;
```

```
 CheckBox8->Enabled = false;
 CheckBox8->Checked = false;
 CheckBox9->Enabled = false;
 CheckBox9->Checked = false;
 CheckBox10->Enabled = false;
 CheckBox10->Checked = false;
 }
}
//---------------------------------------------------------------------------

void __fastcall TF_Main::CheckBox7Click(TObject *Sender)
{
Edit_Baut7->Enabled=true;
if (CheckBox7->Checked) CheckBox8->Enabled = true;
else
 {
 CheckBox8->Enabled = false;
 CheckBox8->Checked = false;
 CheckBox9->Enabled = false;
 CheckBox9->Checked = false;
 CheckBox10->Enabled = false;
 CheckBox10->Checked = false;
 }
}
//---------------------------------------------------------------------------

void __fastcall TF_Main::CheckBox8Click(TObject *Sender)
{
Edit_Baut8->Enabled=true;
if (CheckBox8->Checked) CheckBox9->Enabled = true;
else
 {
 CheckBox9->Enabled = false;
 CheckBox9->Checked = false;
 CheckBox10->Enabled = false;
 CheckBox10->Checked = false;
 }
}
//---------------------------------------------------------------------------

void __fastcall TF_Main::CheckBox9Click(TObject *Sender)
{
Edit_Baut9->Enabled=true;
if (CheckBox9->Checked) CheckBox10->Enabled = true;
else
 {
 CheckBox10->Enabled = false;
 CheckBox10->Checked = false;
 }
}
//---------------------------------------------------------------------------

void __fastcall TF_Main::CheckBox10Click(TObject *Sender)
{
Edit_Baut10->Enabled=true;
}
//---------------------------------------------------------------------------
// RadioGroup definieren, nach Konfiguration der Bauteile
//---------------------------------------------------------------------------
```

```
void __fastcall TF_Main::Button7Click(TObject *Sender)
{
RadioGroup1->Items->Clear();

if (CheckBox1->Checked)
{
TabSheet_Baut1->TabVisible = true;
RadioGroup1->Items->Add(Edit_Baut1->Text);
}
else
{
TabSheet_Baut1->TabVisible = false;
}
TabSheet_Baut1->Caption = Edit_Baut1->Text;
if (CheckBox2->Checked)
{
TabSheet_Baut2->TabVisible = true;
RadioGroup1->Items->Add(Edit_Baut2->Text);
}
else
{
TabSheet_Baut2->TabVisible = false;
}
TabSheet_Baut2->Caption = Edit_Baut2->Text;
if (CheckBox3->Checked)
{
TabSheet_Baut3->TabVisible = true;
RadioGroup1->Items->Add(Edit_Baut3->Text);
}
else
{
TabSheet_Baut3->TabVisible = false;
}
TabSheet_Baut3->Caption = Edit_Baut3->Text;
if (CheckBox4->Checked)
{
TabSheet_Baut4->TabVisible = true;
RadioGroup1->Items->Add(Edit_Baut4->Text);
}
else
{
TabSheet_Baut4->TabVisible = false;
}
TabSheet_Baut4->Caption = Edit_Baut4->Text;
if (CheckBox5->Checked)
{
TabSheet_Baut5->TabVisible = true;
RadioGroup1->Items->Add(Edit_Baut5->Text);
}
else
{
TabSheet_Baut5->TabVisible = false;
}
TabSheet_Baut5->Caption = Edit_Baut5->Text;
if (CheckBox6->Checked)
{
TabSheet_Baut6->TabVisible = true;
RadioGroup1->Items->Add(Edit_Baut6->Text);
}
else
```

```
{
TabSheet_Baut6->TabVisible = false;
}
TabSheet_Baut6->Caption = Edit_Baut6->Text;
if (CheckBox7->Checked)
{
TabSheet_Baut7->TabVisible = true;
RadioGroup1->Items->Add(Edit_Baut7->Text);
}
else
{
TabSheet_Baut7->TabVisible = false;
}
TabSheet_Baut7->Caption = Edit_Baut7->Text;
if (CheckBox8->Checked)
{
TabSheet_Baut8->TabVisible = true;
RadioGroup1->Items->Add(Edit_Baut8->Text);
}
else
{
TabSheet_Baut8->TabVisible = false;
}
TabSheet_Baut8->Caption = Edit_Baut8->Text;
if (CheckBox9->Checked)
{
TabSheet_Baut9->TabVisible = true;
RadioGroup1->Items->Add(Edit_Baut9->Text);
}
else
{
TabSheet_Baut9->TabVisible = false;
}
TabSheet_Baut9->Caption = Edit_Baut9->Text;
if (CheckBox10->Checked)
{
TabSheet_Baut10->TabVisible = true;
RadioGroup1->Items->Add(Edit_Baut10->Text);
}
else
{
TabSheet_Baut10->TabVisible = false;
}
TabSheet_Baut10->Caption = Edit_Baut10->Text;

RadioGroup1->ItemIndex = 0;
Button7->Enabled = false;
GroupBox1->Enabled = false;
CheckBox1->Enabled = false;
CheckBox2->Enabled = false;
CheckBox3->Enabled = false;
CheckBox4->Enabled = false;
CheckBox5->Enabled = false;
CheckBox6->Enabled = false;
CheckBox7->Enabled = false;
CheckBox8->Enabled = false;
CheckBox9->Enabled = false;
CheckBox10->Enabled = false;
}
//---------------------------------------------------------------------------
```

```
// Speichern der StringGrid1 in verdeckte StringGrids + laden
//---------------------------------------------------------------------------

void __fastcall TF_Main::RadioGroup1Click(TObject *Sender)
{
int iRow, iCol;
//Speichern der Information in eines der 10 verdeckten StringGrids
switch (RadioButtonBautSav)
{
 case 0 :
   // Bauteil 1 abgewählt
   for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
    {
    for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
     {
      pStringGrid_Baut1->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
     }
    }
   break;
 case 1 :
   // Bauteil 2 abgewählt
   for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
    {
    for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
     {
      pStringGrid_Baut2->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
     }
    }
   break;
 case 2 :
   // Bauteil 3 abgewählt
   for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
    {
    for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
     {
      pStringGrid_Baut3->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
     }
    }
   break;
 case 3 :
   // Bauteil 4 abgewählt
   for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
    {
    for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
     {
      pStringGrid_Baut4->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
     }
    }
   break;
 case 4 :
   // Bauteil 5 abgewählt
   for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
    {
    for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
     {
      pStringGrid_Baut5->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
     }
    }
   break;
 case 5 :
```

```
    // Bauteil 6 abgewählt
    for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
     {
     for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
      {
       pStringGrid_Baut6->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
      }
     }
    break;
  case 6 :
    // Bauteil 7 abgewählt
    for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
     {
     for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
      {
       pStringGrid_Baut7->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
      }
     }
    break;
  case 7 :
    // Bauteil 8 abgewählt
    for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
     {
     for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
      {
       pStringGrid_Baut8->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
      }
     }
    break;
  case 8 :
    // Bauteil 9 abgewählt
    for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
     {
     for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
      {
       pStringGrid_Baut9->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
      }
     }
    break;
  case 9 :
    // Bauteil 10 abgewählt
    for (iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
     {
     for (iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
      {
       pStringGrid_Baut10->Cells[iCol][iRow] = StringGrid1->Cells[iCol][iRow];
      }
     }
    break;
}


//Holen der Information aus einem der 10 verdeckten StringGrids in das sichtbare
switch (RadioGroup1->ItemIndex)
{
  case 0 :
    // Bauteil 1 ausgewählt
    for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
     {
     for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
```

```
    {
     StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut1->Cells[iCol][iRow];
    }
   }
  break;
 case 1 :
  //  if (pStringGrid_Baut2->Cells[0][0] == "") break;
   // Bauteil 2 ausgewählt
   for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
    {
    for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
     {
      StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut2->Cells[iCol][iRow];
     }
    }
  break;
 case 2 :
    if (pStringGrid_Baut3->Cells[0][0] == "") break;
   // Bauteil 3 ausgewählt
   for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
    {
    for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
     {
      StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut3->Cells[iCol][iRow];
     }
    }
  break;
 case 3 :
    if (pStringGrid_Baut4->Cells[0][0] == "") break;
   // Bauteil 4 ausgewählt
   for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
    {
    for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
     {
      StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut4->Cells[iCol][iRow];
     }
    }
  break;
 case 4 :
   // Bauteil 5 ausgewählt
    if (pStringGrid_Baut5->Cells[0][0] == "") break;
   for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
    {
    for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
     {
      StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut5->Cells[iCol][iRow];
     }
    }
  break;
 case 5 :
    if (pStringGrid_Baut6->Cells[0][0] == "") break;
   // Bauteil 6 ausgewählt
   for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
    {
    for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
     {
      StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut6->Cells[iCol][iRow];
     }
    }
  break;
```

```
  case 6 :
    if (pStringGrid_Baut7->Cells[0][0] == "") break;
   // Bauteil 7 ausgewählt
   for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
    {
    for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
     {
      StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut7->Cells[iCol][iRow];
     }
    }
   break;
  case 7 :
    if (pStringGrid_Baut8->Cells[0][0] == "") break;
   // Bauteil 8 ausgewählt
   for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
    {
    for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
     {
      StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut8->Cells[iCol][iRow];
     }
    }
   break;
  case 8 :
    if (pStringGrid_Baut9->Cells[0][0] == "") break;
   // Bauteil 9 ausgewählt
   for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
    {
    for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
     {
      StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut9->Cells[iCol][iRow];
     }
    }
   break;
  case 9 :
    if (pStringGrid_Baut10->Cells[0][0] == "") break;
   // Bauteil 10 ausgewählt
   for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
    {
    for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
     {
      StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut10->Cells[iCol][iRow];
     }
    }
   break;
}
//aktuellen ausgewählten RadioButton zuweisen
RadioButtonBautSav = RadioGroup1->ItemIndex;

}
//---------------------------------------------------------------------------
// Aufruf Rabattblatt
//---------------------------------------------------------------------------

void TF_Main::AufrufRabattblatt(void)
{
}
//---------------------------------------------------------------------------
// Reset des ganzen Projekts (löschen generierter Daten)
//---------------------------------------------------------------------------
```

```
void __fastcall TF_Main::Reset1Click(TObject *Sender)
{
RadioGroup1->Items->Clear();
RadioGroup1->Items->Add("Bauteil 1");
TabSheet_Baut1->TabVisible = false;
TabSheet_Baut2->TabVisible = false;
TabSheet_Baut3->TabVisible = false;
TabSheet_Baut4->TabVisible = false;
TabSheet_Baut5->TabVisible = false;
TabSheet_Baut6->TabVisible = false;
TabSheet_Baut7->TabVisible = false;
TabSheet_Baut8->TabVisible = false;
TabSheet_Baut9->TabVisible = false;
TabSheet_Baut10->TabVisible = false;

RadioGroup1->ItemIndex = 0;
Button7->Enabled = true;
GroupBox1->Enabled = true;
CheckBox1->Enabled = true;
CheckBox1->Checked = false;
CheckBox2->Enabled = false;
CheckBox3->Enabled = false;
CheckBox4->Enabled = false;
CheckBox5->Enabled = false;
CheckBox6->Enabled = false;
CheckBox7->Enabled = false;
CheckBox8->Enabled = false;
CheckBox9->Enabled = false;
CheckBox10->Enabled = false;

for (int ii = 1; ii <= 11; ii++)
 for (int i = 1; i <= AnzahlZeilen; i++)
   for (int j = 0; j <= 26; j++)
    switch (ii)
    {
    case  1 : pStringGrid_Baut1->Cells[j][i] = "";
              break;
    case  2 : pStringGrid_Baut2->Cells[j][i] = "";
              break;
    case  3 : pStringGrid_Baut3->Cells[j][i] = "";
              break;
    case  4 : pStringGrid_Baut4->Cells[j][i] = "";
              break;
    case  5 : pStringGrid_Baut5->Cells[j][i] = "";
              break;
    case  6 : pStringGrid_Baut6->Cells[j][i] = "";
              break;
    case  7 : pStringGrid_Baut7->Cells[j][i] = "";
              break;
    case  8 : pStringGrid_Baut8->Cells[j][i] = "";
              break;
    case  9 : pStringGrid_Baut9->Cells[j][i] = "";
              break;
    case  10: pStringGrid_Baut10->Cells[j][i] = "";
              break;
    case  11: StringGrid1->Cells[j][i] = "";
              break;
    };
//löschen der Kalkulation
pKalkulation_Baut1->ClearKalk();
```

```
pKalkulation_Baut2->ClearKalk();
pKalkulation_Baut3->ClearKalk();
pKalkulation_Baut4->ClearKalk();
pKalkulation_Baut5->ClearKalk();
pKalkulation_Baut6->ClearKalk();
pKalkulation_Baut7->ClearKalk();
pKalkulation_Baut8->ClearKalk();
pKalkulation_Baut9->ClearKalk();
pKalkulation_Baut10->ClearKalk();

//löschen StringGrid_GLT
for (int i = 1 ; i <=StringGrid_GLT->RowCount; i++)
   for (int j = 0 ; j <=StringGrid_GLT->ColCount; j++)
      StringGrid_GLT->Cells[j][i] = "";

}
//---------------------------------------------------------------------------
// Speichern der Änderung in EK Rabattblatt
//---------------------------------------------------------------------------

void __fastcall TF_Main::Rabatt_EK_SpeichernClick(TObject *Sender)
{
//Speichern der Änderung in EK in ExtraDatei Rabatt.blatt
 TStringList* myList = new TStringList();

 myList->Add(Edit_SW_EK->Text);
 for (int h=1; h < StringGrid_EK->RowCount; h++)
  {
   myList->Add(StringGrid_EK->Cells[0][h]);
   myList->Add(StringGrid_EK->Cells[1][h]);
   myList->Add(StringGrid_EK->Cells[2][h]);
  }

 myList->SaveToFile(Dir+"\\files\\Rabatt.blatt");
 delete myList;
}
//---------------------------------------------------------------------------
// Neue Zeile in EK Rabattblatt einfügen
//---------------------------------------------------------------------------

void __fastcall TF_Main::Neue_Zeile_Rabatt_EKClick(TObject *Sender)
{
//Neue Zeile hinzufügen
StringGrid_EK->RowCount++;
}
//---------------------------------------------------------------------------
// löschen selektierter Zeile in EK Rabattblatt
//---------------------------------------------------------------------------

void __fastcall TF_Main::Zeile_Loeschen_Rabatt_EKClick(TObject *Sender)
{
//Wenn eine Zeile ausgewählt, dann diese löschen
if (delRow_EK !=0)
{
  for (int i = delRow_EK; i<=StringGrid_EK->RowCount;i++)
    {
      StringGrid_EK->Cells[0][i] = StringGrid_EK->Cells[0][i+1];
      StringGrid_EK->Cells[1][i] = StringGrid_EK->Cells[1][i+1];
      StringGrid_EK->Cells[2][i] = StringGrid_EK->Cells[2][i+1];
    }
```

```
  StringGrid_EK->Cells[0][StringGrid_EK->RowCount-1] = "";
  StringGrid_EK->Cells[1][StringGrid_EK->RowCount-1] = "";
  StringGrid_EK->Cells[2][StringGrid_EK->RowCount-1] = "";
  StringGrid_EK->RowCount--;
}
else ShowMessage ("Keine Zeile ausgewählt");
}
//---------------------------------------------------------------------------
// selektierte Zeile von EK Rabattblatt in globale Var speichern
//---------------------------------------------------------------------------

void __fastcall TF_Main::StringGrid_EKSelectCell(TObject *Sender, int ACol,
      int ARow, bool &CanSelect)
{
//Selektierte Zelle (Zeile) in glob. Var speichern
delRow_EK = ARow;
}
//---------------------------------------------------------------------------
// Neue Zeile in EK Rabattblatt einfügen
//---------------------------------------------------------------------------

void __fastcall TF_Main::Neue_Zeile_Rabatt_VKClick(TObject *Sender)
{
//Neue Zeile hinzufügen
StringGrid_VK->RowCount++;
AenderungDurchgefuhert = true;
}
//---------------------------------------------------------------------------
// löschen selektierter Zeile in EK Rabattblatt
//---------------------------------------------------------------------------

void __fastcall TF_Main::Zeile_Loeschen_Rabatt_VKClick(TObject *Sender)
{
//Wenn eine Zeile ausgewählt, dann diese löschen
if (delRow_VK !=0)
{
  for (int i = delRow_VK; i<=StringGrid_VK->RowCount;i++)
    {
      StringGrid_VK->Cells[0][i] = StringGrid_VK->Cells[0][i+1];
      StringGrid_VK->Cells[1][i] = StringGrid_VK->Cells[1][i+1];
      StringGrid_VK->Cells[2][i] = StringGrid_VK->Cells[2][i+1];
    }

  StringGrid_VK->Cells[0][StringGrid_VK->RowCount-1] = "";
  StringGrid_VK->Cells[1][StringGrid_VK->RowCount-1] = "";
  StringGrid_VK->Cells[2][StringGrid_VK->RowCount-1] = "";
  StringGrid_VK->RowCount--;
  AenderungDurchgefuhert = true;
}
else ShowMessage ("Keine Zeile ausgewählt");
}
//---------------------------------------------------------------------------
// selektierte Zeile von VK Rabattblatt in globale Var speichern
//---------------------------------------------------------------------------

void __fastcall TF_Main::StringGrid_VKSelectCell(TObject *Sender, int ACol,
      int ARow, bool &CanSelect)
{
//Selektierte Zelle(Zeile) in glob. Var speichern
```

```
delRow_VK = ARow;
}
//---------------------------------------------------------------------------
// Datenübernahme in Kalkulation
//---------------------------------------------------------------------------

void __fastcall TF_Main::DatenuebernahmeClick(TObject *Sender)
{
//löschen der Kalkulation
pKalkulation_Baut1->ClearKalk();
pKalkulation_Baut2->ClearKalk();
pKalkulation_Baut3->ClearKalk();
pKalkulation_Baut4->ClearKalk();
pKalkulation_Baut5->ClearKalk();
pKalkulation_Baut6->ClearKalk();
pKalkulation_Baut7->ClearKalk();
pKalkulation_Baut8->ClearKalk();
pKalkulation_Baut9->ClearKalk();
pKalkulation_Baut10->ClearKalk();

//löschen StringGrid_GLT
for (int i = 1 ; i <=StringGrid_GLT->RowCount; i++)
  for (int j = 0 ; j <=StringGrid_GLT->ColCount; j++)
     StringGrid_GLT->Cells[j][i] = "";

StringGrid_GLT->RowCount = 2;

//Daten in Kalkulation schreiben
pKalkulation_Baut1->Datenuebergabe(pStringGrid_Baut1);
pKalkulation_Baut2->Datenuebergabe(pStringGrid_Baut2);
pKalkulation_Baut3->Datenuebergabe(pStringGrid_Baut3);
pKalkulation_Baut4->Datenuebergabe(pStringGrid_Baut4);
pKalkulation_Baut5->Datenuebergabe(pStringGrid_Baut5);
pKalkulation_Baut6->Datenuebergabe(pStringGrid_Baut6);
pKalkulation_Baut7->Datenuebergabe(pStringGrid_Baut7);
pKalkulation_Baut8->Datenuebergabe(pStringGrid_Baut8);
pKalkulation_Baut9->Datenuebergabe(pStringGrid_Baut9);
pKalkulation_Baut10->Datenuebergabe(pStringGrid_Baut10);

//StringGrid GLT mit Daten befüllen mittls rekursivem Aufruf
int FuellstandGLT =
FillGLT(pStringGrid_Baut10,Edit_Baut10->Text,
FillGLT(pStringGrid_Baut9,Edit_Baut9->Text,
FillGLT(pStringGrid_Baut8,Edit_Baut8->Text,
FillGLT(pStringGrid_Baut7,Edit_Baut7->Text,
FillGLT(pStringGrid_Baut6,Edit_Baut6->Text,
FillGLT(pStringGrid_Baut5,Edit_Baut5->Text,
FillGLT(pStringGrid_Baut4,Edit_Baut4->Text,
FillGLT(pStringGrid_Baut3,Edit_Baut3->Text,
FillGLT(pStringGrid_Baut2,Edit_Baut2->Text,
FillGLT(pStringGrid_Baut1,Edit_Baut1->Text,0)))))))));
if (StringGrid_GLT->RowCount >2) StringGrid_GLT->RowCount--;

}
//---------------------------------------------------------------------------
// Test: Eingabe in Editfeld in Zahlenformat ändern
//---------------------------------------------------------------------------

void __fastcall TF_Main::SW_EuroChange(TObject *Sender)
{
```

```
 SW_IBS_Ges_Euro->Text                =FloatToStrF(StrToFloat(SW_Euro->Text)+StrToFloat(Doku_Euro-
>Text)+StrToFloat(IBN_Euro->Text)+StrToFloat(Reisekosten_Euro->Text),ffFixed,8,2);
 SW_IBS_Ges_Euro_EK->Text  =  FloatToStrF(StrToFloat(SW_Euro_EK->Text)+StrToFloat(Doku_Euro_EK-
>Text)+StrToFloat(IBN_Euro_EK->Text)+StrToFloat(Reisekosten_Euro_EK->Text),ffFixed,8,2);
}
//---------------------------------------------------------------------------
void __fastcall TF_Main::SW_Pro_DatenpktChange(TObject *Sender)
{
try
 {
 SW_Pro_Datenpkt->Text = FloatToStrF(StrToFloat(SW_Pro_Datenpkt->Text),ffFixed,8,2);
 History_SW_Pro_Datenpkt = SW_Pro_Datenpkt->Text;
 SW_Euro->Text          =          FloatToStrF(StrToInt(Datenpunkte->Text)*StrToFloat(SW_Pro_Datenpkt-
>Text),ffFixed,8,2);
 SW_Euro_EK->Text      =      FloatToStrF(StrToInt(Datenpunkte->Text)*StrToFloat(SW_Pro_Datenpkt_EK-
>Text),ffFixed,8,2);
 }
 catch(...)
 {
 ShowMessage("Bitte geben sie einen Zahlenwert ein!!!");
 SW_Pro_Datenpkt->Text = History_SW_Pro_Datenpkt;
 }
}
//---------------------------------------------------------------------------

void __fastcall TF_Main::SW_Pro_Datenpkt_EKChange(TObject *Sender)
{
try
 {
 SW_Pro_Datenpkt_EK->Text = FloatToStrF(StrToFloat(SW_Pro_Datenpkt_EK->Text),ffFixed,8,2);
 History_SW_Pro_Datenpkt_EK = SW_Pro_Datenpkt_EK->Text;
 SW_Euro->Text          =          FloatToStrF(StrToInt(Datenpunkte->Text)*StrToFloat(SW_Pro_Datenpkt-
>Text),ffFixed,8,2);
 SW_Euro_EK->Text      =      FloatToStrF(StrToInt(Datenpunkte->Text)*StrToFloat(SW_Pro_Datenpkt_EK-
>Text),ffFixed,8,2);
 }
catch(...)
 {
 ShowMessage("Bitte geben sie einen Zahlenwert ein!!!");
 SW_Pro_Datenpkt_EK->Text = History_SW_Pro_Datenpkt_EK;
 }
}
//---------------------------------------------------------------------------
void __fastcall TF_Main::Doku_Pro_DatenpktChange(TObject *Sender)
{
try
 {
 Doku_Pro_Datenpkt->Text = FloatToStrF(StrToFloat(Doku_Pro_Datenpkt->Text),ffFixed,8,2);
 History_Doku_Pro_Datenpkt = Doku_Pro_Datenpkt->Text;
 Doku_Euro->Text          =          FloatToStrF(StrToInt(Datenpunkte->Text)*StrToFloat(Doku_Pro_Datenpkt-
>Text),ffFixed,8,2);
 Doku_Euro_EK->Text    =    FloatToStrF(StrToInt(Datenpunkte->Text)*StrToFloat(Doku_Pro_Datenpkt_EK-
>Text),ffFixed,8,2);
 }
catch(...)
 {
 ShowMessage("Bitte geben sie einen Zahlenwert ein!!!");
 Doku_Pro_Datenpkt->Text = History_Doku_Pro_Datenpkt;
 }
}
```

```
//---------------------------------------------------------------------------
void __fastcall TF_Main::Doku_Pro_Datenpkt_EKChange(TObject *Sender)
{
try
 {
 Doku_Pro_Datenpkt_EK->Text = FloatToStrF(StrToFloat(Doku_Pro_Datenpkt_EK->Text),ffFixed,8,2);
 History_Doku_Pro_Datenpkt_EK = Doku_Pro_Datenpkt_EK->Text;
 Doku_Euro->Text          =          FloatToStrF(StrToInt(Datenpunkte->Text)*StrToFloat(Doku_Pro_Datenpkt-
>Text),ffFixed,8,2);
 Doku_Euro_EK->Text    =    FloatToStrF(StrToInt(Datenpunkte->Text)*StrToFloat(Doku_Pro_Datenpkt_EK-
>Text),ffFixed,8,2);
 }
catch(...)
 {
 ShowMessage("Bitte geben sie einen Zahlenwert ein!!!");
 Doku_Pro_Datenpkt_EK->Text = History_Doku_Pro_Datenpkt_EK;
 }
}
//---------------------------------------------------------------------------
void __fastcall TF_Main::IBN_Pro_DatenpktChange(TObject *Sender)
{
try
 {
 IBN_Pro_Datenpkt->Text = FloatToStrF(StrToFloat(IBN_Pro_Datenpkt->Text),ffFixed,8,2);
 History_IBN_Pro_Datenpkt = IBN_Pro_Datenpkt->Text;
 IBN_Euro->Text          =          FloatToStrF(StrToInt(Datenpunkte->Text)*StrToFloat(IBN_Pro_Datenpkt-
>Text),ffFixed,8,2);
 IBN_Euro_EK->Text    =    FloatToStrF(StrToInt(Datenpunkte->Text)*StrToFloat(IBN_Pro_Datenpkt_EK-
>Text),ffFixed,8,2);
 }
catch(...)
 {
 ShowMessage("Bitte geben sie einen Zahlenwert ein!!!");
 IBN_Pro_Datenpkt->Text = History_IBN_Pro_Datenpkt;
 }
}
//---------------------------------------------------------------------------
void __fastcall TF_Main::IBN_Pro_Datenpkt_EKChange(TObject *Sender)
{
try
 {
 IBN_Pro_Datenpkt_EK->Text = FloatToStrF(StrToFloat(IBN_Pro_Datenpkt_EK->Text),ffFixed,8,2);
 History_IBN_Pro_Datenpkt_EK = IBN_Pro_Datenpkt_EK->Text;
 IBN_Euro->Text          =          FloatToStrF(StrToInt(Datenpunkte->Text)*StrToFloat(IBN_Pro_Datenpkt-
>Text),ffFixed,8,2);
 IBN_Euro_EK->Text    =    FloatToStrF(StrToInt(Datenpunkte->Text)*StrToFloat(IBN_Pro_Datenpkt_EK-
>Text),ffFixed,8,2);
 }
catch(...)
 {
 ShowMessage("Bitte geben sie einen Zahlenwert ein!!!");
 IBN_Pro_Datenpkt_EK->Text = History_IBN_Pro_Datenpkt_EK;
 }
}
//---------------------------------------------------------------------------
// Test: Eingabe in Editfeld in Zahlenformat ändern
//---------------------------------------------------------------------------

void __fastcall TF_Main::Reise_EuroChange(TObject *Sender)
{
```

```
try
 {
 Reise_Euro->Text = FloatToStrF(StrToFloat(Reise_Euro->Text),ffFixed,8,2);
 History_Reise_Euro = Reise_Euro->Text;
 Reisekosten_Euro->Text =FloatToStrF(StrToInt(Anz_Reise_Euro->Text)*StrToFloat(Reise_Euro-
>Text)+StrToInt(Anz_Hotel_Euro->Text)*StrToFloat(Hotel_Euro->Text),ffFixed,8,2);
 Reisekosten_Euro_EK->Text = FloatToStrF(StrToInt(Anz_Reise_Euro->Text)*StrToFloat(Reise_Euro_EK-
>Text)+StrToInt(Anz_Hotel_Euro->Text)*StrToFloat(Hotel_Euro_EK->Text),ffFixed,8,2);
 }
catch(...)
 {
 ShowMessage("Bitte geben sie einen Zahlenwert ein!!!");
 Reise_Euro->Text = History_Reise_Euro;
 }
}
//---------------------------------------------------------------------------
void __fastcall TF_Main::Reise_Euro_EKChange(TObject *Sender)
{
try
 {
 Reise_Euro_EK->Text = FloatToStrF(StrToFloat(Reise_Euro_EK->Text),ffFixed,8,2);
 History_Reise_Euro_EK = Reise_Euro_EK->Text;
 Reisekosten_Euro->Text =FloatToStrF(StrToInt(Anz_Reise_Euro->Text)*StrToFloat(Reise_Euro-
>Text)+StrToInt(Anz_Hotel_Euro->Text)*StrToFloat(Hotel_Euro->Text),ffFixed,8,2);
 Reisekosten_Euro_EK->Text = FloatToStrF(StrToInt(Anz_Reise_Euro->Text)*StrToFloat(Reise_Euro_EK-
>Text)+StrToInt(Anz_Hotel_Euro->Text)*StrToFloat(Hotel_Euro_EK->Text),ffFixed,8,2);
 }
catch(...)
 {
 ShowMessage("Bitte geben sie einen Zahlenwert ein!!!");
 Reise_Euro_EK->Text = History_Reise_Euro_EK;
 }
}
//---------------------------------------------------------------------------
// Eingabe in Editfeld in Zahlenformat ändern
//---------------------------------------------------------------------------
void __fastcall TF_Main::Hotel_EuroChange(TObject *Sender)
{
try
 {
 Hotel_Euro->Text = FloatToStrF(StrToFloat(Hotel_Euro->Text),ffFixed,8,2);
 History_Hotel_Euro = Hotel_Euro->Text;
 Reisekosten_Euro->Text =FloatToStrF(StrToInt(Anz_Reise_Euro->Text)*StrToFloat(Reise_Euro-
>Text)+StrToInt(Anz_Hotel_Euro->Text)*StrToFloat(Hotel_Euro->Text),ffFixed,8,2);
 Reisekosten_Euro_EK->Text = FloatToStrF(StrToInt(Anz_Reise_Euro->Text)*StrToFloat(Reise_Euro_EK-
>Text)+StrToInt(Anz_Hotel_Euro->Text)*StrToFloat(Hotel_Euro_EK->Text),ffFixed,8,2);
 }
catch(...)
 {
 ShowMessage("Bitte geben sie einen Zahlenwert ein!!!");
 Hotel_Euro->Text = History_Hotel_Euro;
 }
}
//---------------------------------------------------------------------------

void __fastcall TF_Main::Hotel_Euro_EKChange(TObject *Sender)
{
try
 {
 Hotel_Euro_EK->Text = FloatToStrF(StrToFloat(Hotel_Euro_EK->Text),ffFixed,8,2);
```

```
 History_Hotel_Euro_EK = Hotel_Euro_EK->Text;
 Reisekosten_Euro->Text                =FloatToStrF(StrToInt(Anz_Reise_Euro->Text)*StrToFloat(Reise_Euro-
>Text)+StrToInt(Anz_Hotel_Euro->Text)*StrToFloat(Hotel_Euro->Text),ffFixed,8,2);
 Reisekosten_Euro_EK->Text   =   FloatToStrF(StrToInt(Anz_Reise_Euro->Text)*StrToFloat(Reise_Euro_EK-
>Text)+StrToInt(Anz_Hotel_Euro->Text)*StrToFloat(Hotel_Euro_EK->Text),ffFixed,8,2);
 }
catch(...)
 {
 ShowMessage("Bitte geben sie einen Zahlenwert ein!!!");
 Hotel_Euro_EK->Text = History_Hotel_Euro_EK;
 }
}
//---------------------------------------------------------------------------
// Neuberechnung Gesamt
//---------------------------------------------------------------------------
void __fastcall TF_Main::Anz_Reise_EuroChange(TObject *Sender)
{
 Reisekosten_Euro->Text                =FloatToStrF(StrToInt(Anz_Reise_Euro->Text)*StrToFloat(Reise_Euro-
>Text)+StrToInt(Anz_Hotel_Euro->Text)*StrToFloat(Hotel_Euro->Text),ffFixed,8,2);
 Reisekosten_Euro_EK->Text   =   FloatToStrF(StrToInt(Anz_Reise_Euro->Text)*StrToFloat(Reise_Euro_EK-
>Text)+StrToInt(Anz_Hotel_Euro->Text)*StrToFloat(Hotel_Euro_EK->Text),ffFixed,8,2);
}
//---------------------------------------------------------------------------
// Neuberechnung Gesamt
//---------------------------------------------------------------------------
void __fastcall TF_Main::Anz_Hotel_EuroChange(TObject *Sender)
{
 Reisekosten_Euro->Text                =FloatToStrF(StrToInt(Anz_Reise_Euro->Text)*StrToFloat(Reise_Euro-
>Text)+StrToInt(Anz_Hotel_Euro->Text)*StrToFloat(Hotel_Euro->Text),ffFixed,8,2);
 Reisekosten_Euro_EK->Text   =   FloatToStrF(StrToInt(Anz_Reise_Euro->Text)*StrToFloat(Reise_Euro_EK-
>Text)+StrToInt(Anz_Hotel_Euro->Text)*StrToFloat(Hotel_Euro_EK->Text),ffFixed,8,2);
}
//---------------------------------------------------------------------------
// Befüllen der GLT für Kalkulation
//---------------------------------------------------------------------------

int __fastcall TF_Main::FillGLT(TStringGrid *StrGd, AnsiString Bauteil, int Index)
{
int StringGrid_GLTRowIndex=Index;
for (int i =1; i <= StrGd->RowCount; i++)
{
  //GLT in StringGrid_GLT übernehmen
  if (StrGd->Cells[25][i] == "GLT" && StrGd->Cells[0][i]  == "OK")
  {
  StringGrid_GLT->RowCount++;
  StringGrid_GLTRowIndex++;
  StringGrid_GLT->Cells[0][StringGrid_GLTRowIndex] = Bauteil;
  StringGrid_GLT->Cells[1][StringGrid_GLTRowIndex] = StrGd->Cells[1][i];
  StringGrid_GLT->Cells[2][StringGrid_GLTRowIndex] = StrGd->Cells[2][i];
  StringGrid_GLT->Cells[3][StringGrid_GLTRowIndex] = StrGd->Cells[12][i];
  StringGrid_GLT->Cells[4][StringGrid_GLTRowIndex] = StrGd->Cells[13][i];
  StringGrid_GLT->Cells[5][StringGrid_GLTRowIndex] = StrGd->Cells[14][i];
  StringGrid_GLT->Cells[6][StringGrid_GLTRowIndex] = StrGd->Cells[19][i];
  StringGrid_GLT->Cells[7][StringGrid_GLTRowIndex] = StrGd->Cells[20][i];
  StringGrid_GLT->Cells[8][StringGrid_GLTRowIndex] = StrGd->Cells[21][i];
  StringGrid_GLT->Cells[9][StringGrid_GLTRowIndex] = StrGd->Cells[22][i];
  StringGrid_GLT->Cells[10][StringGrid_GLTRowIndex] = StrGd->Cells[23][i];
  StringGrid_GLT->Cells[11][StringGrid_GLTRowIndex] = StrGd->Cells[24][i];
  }
}
```

```
return (StringGrid_GLTRowIndex);
}
//---------------------------------------------------------------------------
// Kalkulation durchführen ...
//---------------------------------------------------------------------------

void __fastcall TF_Main::KalkulationClick(TObject *Sender)
{

//Aufruf der Kalkulation
pKalkulation_Baut1->Kalkulation();
pKalkulation_Baut2->Kalkulation();
pKalkulation_Baut3->Kalkulation();
pKalkulation_Baut4->Kalkulation();
pKalkulation_Baut5->Kalkulation();
pKalkulation_Baut6->Kalkulation();
pKalkulation_Baut7->Kalkulation();
pKalkulation_Baut8->Kalkulation();
pKalkulation_Baut9->Kalkulation();
pKalkulation_Baut10->Kalkulation();

float Summe_List1=0,Summe_List2=0,Summe_List3=0;
float Summe_List1netto=0,Summe_List2netto=0,Summe_List3netto=0;
float Summe_List1nettoEK=0,Summe_List2nettoEK=0,Summe_List3nettoEK=0;

int Datenpkt =
pKalkulation_Baut1->Datenpkt +
pKalkulation_Baut2->Datenpkt +
pKalkulation_Baut3->Datenpkt +
pKalkulation_Baut4->Datenpkt +
pKalkulation_Baut5->Datenpkt +
pKalkulation_Baut6->Datenpkt +
pKalkulation_Baut7->Datenpkt +
pKalkulation_Baut8->Datenpkt +
pKalkulation_Baut9->Datenpkt +
pKalkulation_Baut10->Datenpkt;

  if (Datenpkt > 0)
  {
  Table1->Open();
  Query1->Close();
  Query1->SQL->Clear();
  AnsiString ASSQLCommand = "SELECT * FROM Angebot_DB2_Aktuell WHERE Bezeichnung LIKE
'%EBI%' ORDER BY ID";
  Query1->SQL->Add(ASSQLCommand);
  Query1->Open();
  Query1->Active = true;
  Query1->First();
  for (int i=0;i<=20;i++)
   {
   if (Datenpkt < Query1->FieldByName("DatenpktXLWeb")->AsInteger)
     break;
   Query1->Next();
   }
  if (StringGrid_GLT->Cells[1][1] != "") StringGrid_GLT->RowCount++;
  StringGrid_GLT->Cells[0][StringGrid_GLT->RowCount-1] = "1";
  StringGrid_GLT->Cells[1][StringGrid_GLT->RowCount-1] = "1";
  StringGrid_GLT->Cells[2][StringGrid_GLT->RowCount-1]     =     Query1->FieldByName("Bezeichnung")-
>AsString;
```

```
  StringGrid_GLT->Cells[3][StringGrid_GLT->RowCount-1]      =      Query1->FieldByName("Gerätetyp")-
>AsString;
  StringGrid_GLT->Cells[4][StringGrid_GLT->RowCount-1]      =      Query1->FieldByName("Gerätetyp2")-
>AsString;
  StringGrid_GLT->Cells[5][StringGrid_GLT->RowCount-1]      =      Query1->FieldByName("Gerätetyp3")-
>AsString;
  StringGrid_GLT->Cells[6][StringGrid_GLT->RowCount-1]      =      Query1->FieldByName("Hersteller")-
>AsString;
  StringGrid_GLT->Cells[7][StringGrid_GLT->RowCount-1]  =  Query1->FieldByName("RabattgruppeEK")-
>AsString;
  StringGrid_GLT->Cells[8][StringGrid_GLT->RowCount-1]  =  Query1->FieldByName("RabattgruppeVK")-
>AsString;
  StringGrid_GLT->Cells[9][StringGrid_GLT->RowCount-1]  =  Query1->FieldByName("ListenpreisType1")-
>AsString;
  StringGrid_GLT->Cells[10][StringGrid_GLT->RowCount-1] = Query1->FieldByName("ListenpreisType2")-
>AsString;
  StringGrid_GLT->Cells[11][StringGrid_GLT->RowCount-1] = Query1->FieldByName("ListenpreisType3")-
>AsString;
  }

if (StringGrid_GLT->Cells[1][1] != "")
 {
 for (int ii=1;ii<=StringGrid_GLT->RowCount;ii++)
  if (StringGrid_GLT->Cells[1][ii] == "Gesamt")
   {
   //löschen ab "Gesamt" StringGrid_GLT
   for (int i = ii ; i <=StringGrid_GLT->RowCount; i++)
     for (int j = 0 ; j <=StringGrid_GLT->ColCount; j++)
        StringGrid_GLT->Cells[j][i] = "";
   StringGrid_GLT->RowCount = ii;
   }

 for (int i=1;i<=StringGrid_GLT->RowCount-1;i++)
  {
  Summe_List1   =   Summe_List1   +   (StrToFloat(StringGrid_GLT->Cells[9][i])*StrToInt(StringGrid_GLT-
>Cells[1][i]));
  Summe_List2   =   Summe_List2   +   (StrToFloat(StringGrid_GLT->Cells[10][i])*StrToInt(StringGrid_GLT-
>Cells[1][i]));
  Summe_List3   =   Summe_List3   +   (StrToFloat(StringGrid_GLT->Cells[11][i])*StrToInt(StringGrid_GLT-
>Cells[1][i]));
  Summe_List1netto          =          Summe_List1netto          +          (StrToFloat(StringGrid_GLT-
>Cells[9][i])*StrToInt(StringGrid_GLT->Cells[1][i])*Rabkalk(StringGrid_GLT->Cells[8][i]));
  Summe_List2netto          =          Summe_List2netto          +          (StrToFloat(StringGrid_GLT-
>Cells[10][i])*StrToInt(StringGrid_GLT->Cells[1][i])*Rabkalk(StringGrid_GLT->Cells[8][i]));
  Summe_List3netto          =          Summe_List3netto          +          (StrToFloat(StringGrid_GLT-
>Cells[11][i])*StrToInt(StringGrid_GLT->Cells[1][i])*Rabkalk(StringGrid_GLT->Cells[8][i]));
  Summe_List1nettoEK        =        Summe_List1nettoEK        +        (StrToFloat(StringGrid_GLT-
>Cells[9][i])*StrToInt(StringGrid_GLT->Cells[1][i])*RabkalkEK(StringGrid_GLT->Cells[7][i]));
  Summe_List2nettoEK        =        Summe_List2nettoEK        +        (StrToFloat(StringGrid_GLT-
>Cells[10][i])*StrToInt(StringGrid_GLT->Cells[1][i])*RabkalkEK(StringGrid_GLT->Cells[7][i]));
  Summe_List3nettoEK        =        Summe_List3nettoEK        +        (StrToFloat(StringGrid_GLT-
>Cells[11][i])*StrToInt(StringGrid_GLT->Cells[1][i])*RabkalkEK(StringGrid_GLT->Cells[7][i]));
  }

 //Anzeige der Summe
 StringGrid_GLT->RowCount++;
 StringGrid_GLT->Cells[1][StringGrid_GLT->RowCount-1] = "Gesamt";

 StringGrid_GLT->Cells[9][StringGrid_GLT->RowCount-1] = FloatToStrF(Summe_List1netto,ffFixed,8,2);
 StringGrid_GLT->Cells[10][StringGrid_GLT->RowCount-1] = FloatToStrF(Summe_List2netto,ffFixed,8,2);
```

```
 StringGrid_GLT->Cells[11][StringGrid_GLT->RowCount-1] = FloatToStrF(Summe_List3netto,ffFixed,8,2);
 }
//SW+IBS berechnen nach Datenpktanzahl
Datenpunkte->Text = IntToStr(Datenpkt);
SW_Euro->Text = FloatToStrF(Datenpkt * StrToFloat(SW_Pro_Datenpkt->Text),ffFixed,8,2);
SW_Euro_EK->Text = FloatToStrF(Datenpkt * StrToFloat(SW_Pro_Datenpkt_EK->Text),ffFixed,8,2);
Doku_Euro->Text = FloatToStrF(Datenpkt * StrToFloat(Doku_Pro_Datenpkt->Text),ffFixed,8,2);
Doku_Euro_EK->Text = FloatToStrF(Datenpkt * StrToFloat(Doku_Pro_Datenpkt_EK->Text),ffFixed,8,2);
IBN_Euro->Text = FloatToStrF(Datenpkt * StrToFloat(IBN_Pro_Datenpkt->Text),ffFixed,8,2);
IBN_Euro_EK->Text = FloatToStrF(Datenpkt * StrToFloat(IBN_Pro_Datenpkt_EK->Text),ffFixed,8,2);

//Gesamtblatt aufschalten mit Gesamtkalkulation

//TabSheet_Gesamt->SetFocus(); geht nicht
//PageControl2->
//TabSheet_Gesamt->
AnsiString text,text1;
float Summe_List[3];
float GesamtSumme_List[3] = {0,0,0};
float Bauteil1_Preis1[4] = {0,0,0,0}, Bauteil1_Preis2[4] = {0,0,0,0}, Bauteil1_Preis3[4] = {0,0,0,0},
Bauteil1_Preis4[4] = {0,0,0,0};
float Bauteil2_Preis1[4] = {0,0,0,0}, Bauteil2_Preis2[4] = {0,0,0,0}, Bauteil2_Preis3[4] = {0,0,0,0},
Bauteil2_Preis4[4] = {0,0,0,0};
float Bauteil3_Preis1[4] = {0,0,0,0}, Bauteil3_Preis2[4] = {0,0,0,0}, Bauteil3_Preis3[4] = {0,0,0,0},
Bauteil3_Preis4[4] = {0,0,0,0};
float Bauteil4_Preis1[4] = {0,0,0,0}, Bauteil4_Preis2[4] = {0,0,0,0}, Bauteil4_Preis3[4] = {0,0,0,0},
Bauteil4_Preis4[4] = {0,0,0,0};
RichEdit_Gesamt->Clear();

RichEdit_Gesamt->Paragraph->Alignment = taLeftJustify;
if (CheckBox1->Checked)
 {
 RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style << fsBold;
 RichEdit_Gesamt->Lines->Add(Edit_Baut1->Text);
 RichEdit_Gesamt->Lines->Add("");
 AngebotKalk(pKalkulation_Baut1->StringGrid_FuG,"I. Fühler und Geber:",Bauteil1_Preis1);
 AngebotKalk(pKalkulation_Baut1->StringGrid_DDC,"II. DDC:",Bauteil1_Preis2);
 if (SubangebotSS->Checked == false)
   AngebotKalk(pKalkulation_Baut1->StringGrid_SS,"III. Schaltschrank:",Bauteil1_Preis3);
 else
   {
   Bauteil1_Preis3[0] = 0;
   Bauteil1_Preis3[1] = 0;
   Bauteil1_Preis3[2] = 0;
   Bauteil1_Preis3[3] = 0;
   }
 AngebotKalk(pKalkulation_Baut1->StringGrid_SSBG,"IV. Schaltschrank Beistellgeräte:",Bauteil1_Preis4);
 };

if (CheckBox2->Checked)
 {
 RichEdit_Gesamt->Lines->Add("");
 RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style << fsBold;
 RichEdit_Gesamt->Lines->Add(Edit_Baut2->Text);
 RichEdit_Gesamt->Lines->Add("");
 AngebotKalk(pKalkulation_Baut2->StringGrid_FuG,"I. Fühler und Geber:",Bauteil2_Preis1);
 AngebotKalk(pKalkulation_Baut2->StringGrid_DDC,"II. DDC:",Bauteil2_Preis2);
 if (SubangebotSS->Checked == false)
   AngebotKalk(pKalkulation_Baut2->StringGrid_SS,"III. Schaltschrank:",Bauteil2_Preis3);
 else
```

```
  {
  Bauteil2_Preis3[0] = 0;
  Bauteil2_Preis3[1] = 0;
  Bauteil2_Preis3[2] = 0;
  Bauteil2_Preis3[3] = 0;
  }
 AngebotKalk(pKalkulation_Baut2->StringGrid_SSBG,"IV. Schaltschrank Beistellgeräte:",Bauteil2_Preis4);
 }

if (CheckBox3->Checked)
 {
 RichEdit_Gesamt->Lines->Add("");
 RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style << fsBold;
 RichEdit_Gesamt->Lines->Add(Edit_Baut3->Text);
 RichEdit_Gesamt->Lines->Add("");
 AngebotKalk(pKalkulation_Baut3->StringGrid_FuG,"I. Fühler und Geber:",Bauteil3_Preis1);
 AngebotKalk(pKalkulation_Baut3->StringGrid_DDC,"II. DDC:",Bauteil3_Preis2);
 if (SubangebotSS->Checked == false)
   AngebotKalk(pKalkulation_Baut3->StringGrid_SS,"III. Schaltschrank:",Bauteil3_Preis3);
  else
   {
   Bauteil3_Preis3[0] = 0;
   Bauteil3_Preis3[1] = 0;
   Bauteil3_Preis3[2] = 0;
   Bauteil3_Preis3[3] = 0;
   }
 AngebotKalk(pKalkulation_Baut3->StringGrid_SSBG,"IV. Schaltschrank Beistellgeräte:",Bauteil3_Preis4);
 }

if (CheckBox4->Checked)
 {
 RichEdit_Gesamt->Lines->Add("");
 RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style << fsBold;
 RichEdit_Gesamt->Lines->Add(Edit_Baut4->Text);
 RichEdit_Gesamt->Lines->Add("");
 AngebotKalk(pKalkulation_Baut4->StringGrid_FuG,"I. Fühler und Geber:",Bauteil4_Preis1);
 AngebotKalk(pKalkulation_Baut4->StringGrid_DDC,"II. DDC:",Bauteil4_Preis2);
 if (SubangebotSS->Checked == false)
   AngebotKalk(pKalkulation_Baut4->StringGrid_SS,"III. Schaltschrank:",Bauteil4_Preis3);
  else
   {
   Bauteil4_Preis3[0] = 0;
   Bauteil4_Preis3[1] = 0;
   Bauteil4_Preis3[2] = 0;
   Bauteil4_Preis3[3] = 0;
   }
 AngebotKalk(pKalkulation_Baut4->StringGrid_SSBG,"IV. Schaltschrank Beistellgeräte:",Bauteil4_Preis4);
 }


// Gesamtpreis
RichEdit_Gesamt->Paragraph->Alignment = taLeftJustify;
AnsiString Ueberschrift[3] = {"Listenpreis", "Einkaufspreis", "Gesamtpreis"};
Summe_List[0] = Summe_List1+Summe_List2+Summe_List3;
Summe_List[1] = Summe_List1nettoEK+Summe_List2nettoEK+Summe_List3nettoEK;
Summe_List[2] = Summe_List1netto+Summe_List2netto+Summe_List3netto;


//Überschrift
text="";
```

```
for(int i=text.Length();i<=(111-Ueberschrift[2].Length());i++)
 {
 text=text+" ";
 }
text=text+Ueberschrift[2];

RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style << fsBold;
RichEdit_Gesamt->Lines->Add(text);
RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style >> fsBold;
RichEdit_Gesamt->Lines->Add("");

float AufschlagSSSubangebot = 0;
if ((RadioGroup_Preisauswahl->ItemIndex==0 || RadioGroup_Preisauswahl->ItemIndex==1))
 {
 //Kosten Bauteile aufnehmen
 if (CheckBox1->Checked)
  {
  AngebotZusammenstel(Bauteil1_Preis1[3], Bauteil1_Preis2[3], Bauteil1_Preis3[3], Bauteil1_Preis4[3]);
  text="Gesamtkosten "+Edit_Baut1->Text+":";
//                                    for(int                    i=text.Length();i<=(110-FloatToStrF(pKalkulation_Baut1-
>Summe_List[2],ffNumber,8,2).Length());i++)
  for(int                                                                    i=text.Length();i<=(110-
FloatToStrF(Bauteil1_Preis1[3]+Bauteil1_Preis2[3]+Bauteil1_Preis3[3]+Bauteil1_Preis4[3],ffNumber,8,2).Len
gth());i++)
   {
   text=text+" ";
   }
  text=text                                                                                                +
FloatToStrF(Bauteil1_Preis1[3]+Bauteil1_Preis2[3]+Bauteil1_Preis3[3]+Bauteil1_Preis4[3],ffNumber,8,2);
  RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style << fsBold;
  RichEdit_Gesamt->Lines->Add(text);
  RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style >> fsBold;
  RichEdit_Gesamt->Lines->Add("");
  }
 if (CheckBox2->Checked)
  {
  AngebotZusammenstel(Bauteil2_Preis1[3], Bauteil2_Preis2[3], Bauteil2_Preis3[3], Bauteil2_Preis4[3]);
  text="Gesamtkosten "+Edit_Baut2->Text+":";
  for(int                                                                    i=text.Length();i<=(110-
FloatToStrF(Bauteil2_Preis1[3]+Bauteil2_Preis2[3]+Bauteil2_Preis3[3]+Bauteil2_Preis4[3],ffNumber,8,2).Len
gth());i++)
   {
   text=text+" ";
   }
  text=text                                                                                                +
FloatToStrF(Bauteil2_Preis1[3]+Bauteil2_Preis2[3]+Bauteil2_Preis3[3]+Bauteil2_Preis4[3],ffNumber,8,2);
  RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style << fsBold;
  RichEdit_Gesamt->Lines->Add(text);
  RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style >> fsBold;
  RichEdit_Gesamt->Lines->Add("");
  }
 if (CheckBox3->Checked)
  {
  AngebotZusammenstel(Bauteil3_Preis1[3], Bauteil3_Preis2[3], Bauteil3_Preis3[3], Bauteil3_Preis4[3]);
  text="Gesamtkosten "+Edit_Baut3->Text+":";
  for(int                                                                    i=text.Length();i<=(110-
FloatToStrF(Bauteil3_Preis1[3]+Bauteil3_Preis2[3]+Bauteil3_Preis3[3]+Bauteil3_Preis4[3],ffNumber,8,2).Len
gth());i++)
   {
   text=text+" ";
```

```
  }
  text=text +
FloatToStrF(Bauteil3_Preis1[3]+Bauteil3_Preis2[3]+Bauteil3_Preis3[3]+Bauteil3_Preis4[3],ffNumber,8,2);
  RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style << fsBold;
  RichEdit_Gesamt->Lines->Add(text);
  RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style >> fsBold;
  RichEdit_Gesamt->Lines->Add("");
  }
 if (CheckBox4->Checked)
  {
  AngebotZusammenstel(Bauteil4_Preis1[3], Bauteil4_Preis2[3], Bauteil4_Preis3[3], Bauteil4_Preis4[3]);
  text="Gesamtkosten "+Edit_Baut4->Text+":";
  for(int i=text.Length();i<=(110-
FloatToStrF(Bauteil4_Preis1[3]+Bauteil4_Preis2[3]+Bauteil4_Preis3[3]+Bauteil4_Preis4[3],ffNumber,8,2).Len
gth());i++)
   {
   text=text+" ";
   }
  text=text +
FloatToStrF(Bauteil4_Preis1[3]+Bauteil4_Preis2[3]+Bauteil4_Preis3[3]+Bauteil4_Preis4[3],ffNumber,8,2);
  RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style << fsBold;
  RichEdit_Gesamt->Lines->Add(text);
  RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style >> fsBold;
  RichEdit_Gesamt->Lines->Add("");
  }
 float IBS[3];
 IBS[0]= StrToFloat(SW_IBS_Ges_Euro->Text);
 IBS[1]= StrToFloat(SW_IBS_Ges_Euro_EK->Text);
 IBS[2]= StrToFloat(SW_IBS_Ges_Euro->Text);

 // Kosten für SS Subangebot anzeigen
 if (SubangebotSS->Checked)
  {
  text="Gesamtkosten Schaltschrank Subangebot:";
  AufschlagSSSubangebot = StrToFloat(SS_Subangebot_Euro->Text)*StrToFloat(SS_Aufschlag->Text)/100 +
StrToFloat(SS_Subangebot_Euro->Text);
  for(int i=text.Length();i<=(110-FloatToStrF(AufschlagSSSubangebot,ffNumber,8,2).Length());i++)
   {
   text=text+" ";
   }
  text=text + FloatToStrF(AufschlagSSSubangebot,ffNumber,8,2);

  RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style << fsBold;
  RichEdit_Gesamt->Lines->Add(text);
  RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style >> fsBold;
  RichEdit_Gesamt->Lines->Add("");
  }

 //Kosten GLT aufnehmen
 text="Gesamtkosten GLT:";
 for(int i=text.Length();i<=(110-FloatToStrF(Summe_List[2],ffNumber,8,2).Length());i++)
  {
  text=text+" ";
  }
 text=text + FloatToStrF(Summe_List[2],ffNumber,8,2);

 RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style << fsBold;
 RichEdit_Gesamt->Lines->Add(text);
 RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style >> fsBold;
 RichEdit_Gesamt->Lines->Add("");
```

```
//Kosten IBS aufnehmen
text="Softwareerstellung:";
for(int i=text.Length();i<=(90-FloatToStrF(StrToFloat(SW_Euro->Text),ffNumber,8,2).Length());i++)
 {
 text=text+" ";
 }
text=text + FloatToStrF(StrToFloat(SW_Euro->Text),ffNumber,8,2);
RichEdit_Gesamt->Lines->Add(text);
text="Dokumentation:";
for(int i=text.Length();i<=(90-FloatToStrF(StrToFloat(Doku_Euro->Text),ffNumber,8,2).Length());i++)
 {
 text=text+" ";
 }
text=text + FloatToStrF(StrToFloat(Doku_Euro->Text),ffNumber,8,2);
RichEdit_Gesamt->Lines->Add(text);
text="Inbetriebnahme:";
for(int i=text.Length();i<=(90-FloatToStrF(StrToFloat(IBN_Euro->Text),ffNumber,8,2).Length());i++)
 {
 text=text+" ";
 }
text=text + FloatToStrF(StrToFloat(IBN_Euro->Text),ffNumber,8,2);
RichEdit_Gesamt->Lines->Add(text);
text="Reisekosten:";
for(int i=text.Length();i<=(90-FloatToStrF(StrToFloat(Reisekosten_Euro->Text),ffNumber,8,2).Length());i++)
 {
 text=text+" ";
 }
text=text + FloatToStrF(StrToFloat(Reisekosten_Euro->Text),ffNumber,8,2);
RichEdit_Gesamt->Lines->Add(text);

text="Gesamtkosten IBS:";
for(int i=text.Length();i<=(110-FloatToStrF(IBS[2],ffNumber,8,2).Length());i++)
 {
 text=text+" ";
 }
text=text + FloatToStrF(IBS[2],ffNumber,8,2);
RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style << fsBold;
RichEdit_Gesamt->Lines->Add(text);
RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style >> fsBold;
RichEdit_Gesamt->Lines->Add("");
}

//Gesamtkosten ermitteln
for (int i=2;i>=0;i--)
GesamtSumme_List[i]+=Summe_List[i];
for (int i=2;i>=0;i--)
GesamtSumme_List[i]+=pKalkulation_Baut1->Summe_List[i];
for (int i=2;i>=0;i--)
GesamtSumme_List[i]+=pKalkulation_Baut2->Summe_List[i];
for (int i=2;i>=0;i--)
GesamtSumme_List[i]+=pKalkulation_Baut3->Summe_List[i];
for (int i=2;i>=0;i--)
GesamtSumme_List[i]+=pKalkulation_Baut4->Summe_List[i];
for (int i=2;i>=0;i--)
GesamtSumme_List[i]+=StrToFloat(SW_IBS_Ges_Euro->Text);

GesamtSumme_List[2] = Summe_List[2]+StrToFloat(SW_IBS_Ges_Euro-
>Text)+(StrToFloat(SS_Subangebot_Euro->Text)*StrToFloat(SS_Aufschlag-
>Text)/100)+StrToFloat(SS_Subangebot_Euro->Text)+
```

```
Bauteil1_Preis1[3]+Bauteil1_Preis2[3]+Bauteil1_Preis3[3]+Bauteil1_Preis4[3]+
Bauteil2_Preis1[3]+Bauteil2_Preis2[3]+Bauteil2_Preis3[3]+Bauteil2_Preis4[3]+
Bauteil3_Preis1[3]+Bauteil3_Preis2[3]+Bauteil3_Preis3[3]+Bauteil3_Preis4[3]+
Bauteil4_Preis1[3]+Bauteil4_Preis2[3]+Bauteil4_Preis3[3]+Bauteil4_Preis4[3];

text="Summe Netto:";
for(int i=text.Length();i<=(110-FloatToStrF(GesamtSumme_List[2],ffNumber,8,2).Length());i++)
 {
 text=text+" ";
 }
text=text + FloatToStrF(GesamtSumme_List[2],ffNumber,8,2)+" €";
RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style << fsBold;
RichEdit_Gesamt->Lines->Add(text);
RichEdit_Gesamt->Lines->Add("");
if (KundenRabatt->Text != 0)
 {
 text="Rabatt "+KundenRabatt->Text+"%:";
 for(int i=text.Length();i<=(110-FloatToStrF(GesamtSumme_List[2]/100*StrToFloat(KundenRabatt-
>Text),ffNumber,8,2).Length()-1);i++)
  {
  text=text+" ";
  }
 text=text +"-"+ FloatToStrF(GesamtSumme_List[2]/100*StrToFloat(KundenRabatt->Text),ffNumber,8,2)+"
€";

 RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style << fsBold;
 RichEdit_Gesamt->Lines->Add(text);
 RichEdit_Gesamt->Lines->Add("");

 text="ENDSUMME Netto:";
 for(int i=text.Length();i<=(110-FloatToStrF(GesamtSumme_List[2]-
GesamtSumme_List[2]/100*StrToFloat(KundenRabatt->Text),ffNumber,8,2).Length());i++)
  {
  text=text+" ";
  }
 text=text + FloatToStrF(GesamtSumme_List[2]-GesamtSumme_List[2]/100*StrToFloat(KundenRabatt-
>Text),ffNumber,8,2)+" €";

 RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style << fsBold;
 RichEdit_Gesamt->Lines->Add(text);
 RichEdit_Gesamt->Lines->Add("");
 }

 //Kalkulationsblatt

RichEdit_Kalkulationsblatt->Clear();
RichEdit_Kalkulationsblatt->Paragraph->Alignment = taLeftJustify;

if (CheckBox1->Checked)
 {
 RichEdit_Kalkulationsblatt->SelAttributes->Style = RichEdit_Kalkulationsblatt->SelAttributes->Style <<
fsBold;
 RichEdit_Kalkulationsblatt->Lines->Add(Edit_Baut1->Text);
 RichEdit_Kalkulationsblatt->Lines->Add("");
 AngebotKalkblatt(pKalkulation_Baut1->StringGrid_FuG,"I. Fühler und Geber:");
 AngebotKalkblatt(pKalkulation_Baut1->StringGrid_DDC,"II. DDC:");
 if (SubangebotSS->Checked == false)
  AngebotKalkblatt(pKalkulation_Baut1->StringGrid_SS,"III. Schaltschrank:");
 AngebotKalkblatt(pKalkulation_Baut1->StringGrid_SSBG,"IV. Schaltschrank Beistellgeräte:");
 };
```

```
if (CheckBox2->Checked)
 {
 RichEdit_Kalkulationsblatt->Lines->Add("");
 RichEdit_Kalkulationsblatt->SelAttributes->Style = RichEdit_Kalkulationsblatt->SelAttributes->Style << fsBold;
 RichEdit_Kalkulationsblatt->Lines->Add(Edit_Baut2->Text);
 RichEdit_Kalkulationsblatt->Lines->Add("");
 AngebotKalkblatt(pKalkulation_Baut2->StringGrid_FuG,"I. Fühler und Geber:");
 AngebotKalkblatt(pKalkulation_Baut2->StringGrid_DDC,"II. DDC:");
 if (SubangebotSS->Checked == false)
  AngebotKalkblatt(pKalkulation_Baut2->StringGrid_SS,"III. Schaltschrank:");
 AngebotKalkblatt(pKalkulation_Baut2->StringGrid_SSBG,"IV. Schaltschrank Beistellgeräte:");
 }

if (CheckBox3->Checked)
 {
 RichEdit_Kalkulationsblatt->Lines->Add("");
 RichEdit_Kalkulationsblatt->SelAttributes->Style = RichEdit_Kalkulationsblatt->SelAttributes->Style << fsBold;
 RichEdit_Kalkulationsblatt->Lines->Add(Edit_Baut3->Text);
 RichEdit_Kalkulationsblatt->Lines->Add("");
 AngebotKalkblatt(pKalkulation_Baut3->StringGrid_FuG,"I. Fühler und Geber:");
 AngebotKalkblatt(pKalkulation_Baut3->StringGrid_DDC,"II. DDC:");
 if (SubangebotSS->Checked == false)
  AngebotKalkblatt(pKalkulation_Baut3->StringGrid_SS,"III. Schaltschrank:");
 AngebotKalkblatt(pKalkulation_Baut3->StringGrid_SSBG,"IV. Schaltschrank Beistellgeräte:");
 }

if (CheckBox4->Checked)
 {
 RichEdit_Kalkulationsblatt->Lines->Add("");
 RichEdit_Kalkulationsblatt->SelAttributes->Style = RichEdit_Kalkulationsblatt->SelAttributes->Style << fsBold;
 RichEdit_Kalkulationsblatt->Lines->Add(Edit_Baut4->Text);
 RichEdit_Kalkulationsblatt->Lines->Add("");
 AngebotKalkblatt(pKalkulation_Baut4->StringGrid_FuG,"I. Fühler und Geber:");
 AngebotKalkblatt(pKalkulation_Baut4->StringGrid_DDC,"II. DDC:");
 if (SubangebotSS->Checked == false)
  AngebotKalkblatt(pKalkulation_Baut4->StringGrid_SS,"III. Schaltschrank:");
 AngebotKalkblatt(pKalkulation_Baut4->StringGrid_SSBG,"IV. Schaltschrank Beistellgeräte:");
 }

RichEdit_Kalkulationsblatt->Lines->Add("");
RichEdit_Kalkulationsblatt->SelAttributes->Style = RichEdit_Kalkulationsblatt->SelAttributes->Style << fsBold;
RichEdit_Kalkulationsblatt->Lines->Add("Zusammenstellung");
RichEdit_Kalkulationsblatt->Lines->Add("");
if (CheckBox1->Checked)
 {
 AngebotZusammenstelKalkblatt(Edit_Baut1->Text,Bauteil1_Preis1, Bauteil1_Preis2, Bauteil1_Preis3, Bauteil1_Preis4);
 }
if (CheckBox2->Checked)
 {
 AngebotZusammenstelKalkblatt(Edit_Baut2->Text,Bauteil2_Preis1, Bauteil2_Preis2, Bauteil2_Preis3, Bauteil2_Preis4);
 }
if (CheckBox3->Checked)
 {
```

```
 AngebotZusammenstelKalkblatt(Edit_Baut3->Text,Bauteil3_Preis1,  Bauteil3_Preis2,  Bauteil3_Preis3,
Bauteil3_Preis4);
 }
if (CheckBox4->Checked)
 {
 AngebotZusammenstelKalkblatt(Edit_Baut4->Text,Bauteil4_Preis1,  Bauteil4_Preis2,  Bauteil4_Preis3,
Bauteil4_Preis4);
 }
float Preis[4];
//Schaltschrank Subangebot
if (SubangebotSS->Checked)
 {
 Preis[0]= StrToFloat(SS_Subangebot_Euro->Text);
 Preis[1]= Preis[0];
 Preis[2]=          StrToFloat(SS_Subangebot_Euro->Text)*StrToFloat(SS_Aufschlag->Text)/100          +
StrToFloat(SS_Subangebot_Euro->Text);
 Preis[3]= Preis[2];
 AngebotZusammenstelKalkblattAnzeige("Gesamtkosten Schaltschrank Subangebot:",Preis,true);
 RichEdit_Kalkulationsblatt->Lines->Add("");
 }

//GLT
Preis[0]= Summe_List[1];
Preis[1]= Preis[0];
Preis[2]= Summe_List[2];
Preis[3]= Preis[2];
AngebotZusammenstelKalkblattAnzeige("Gesamtkosten GLT:",Preis,true);
RichEdit_Kalkulationsblatt->Lines->Add("");

//Softwareerstellung:
Preis[0]= StrToFloat(SW_Euro_EK->Text);
Preis[1]= Preis[0];
Preis[2]= StrToFloat(SW_Euro->Text);
Preis[3]= Preis[2];
AngebotZusammenstelKalkblattAnzeige("Softwareerstellung:",Preis,false);
//Dokumentation:
Preis[0]= StrToFloat(Doku_Euro_EK->Text);
Preis[1]= Preis[0];
Preis[2]= StrToFloat(Doku_Euro->Text);
Preis[3]= Preis[2];
AngebotZusammenstelKalkblattAnzeige("Dokumentation:",Preis,false);
//Inbetriebnahme:
Preis[0]= StrToFloat(IBN_Euro_EK->Text);
Preis[1]= Preis[0];
Preis[2]= StrToFloat(IBN_Euro->Text);
Preis[3]= Preis[2];
AngebotZusammenstelKalkblattAnzeige("Inbetriebnahme:",Preis,false);
//Reisekosten:
Preis[0]= StrToFloat(Reisekosten_Euro_EK->Text);
Preis[1]= Preis[0];
Preis[2]= StrToFloat(Reisekosten_Euro->Text);
Preis[3]= Preis[2];
AngebotZusammenstelKalkblattAnzeige("Reisekosten:",Preis,false);
RichEdit_Kalkulationsblatt->Lines->Add("");
//SW+IBS:
Preis[0]= StrToFloat(SW_IBS_Ges_Euro_EK->Text);
Preis[1]= Preis[0];
Preis[2]= StrToFloat(SW_IBS_Ges_Euro->Text);
Preis[3]= Preis[2];
AngebotZusammenstelKalkblattAnzeige("Gesamt SW + IBS:",Preis,true);
```

```
RichEdit_Kalkulationsblatt->Lines->Add("");

Preis[0]=          Summe_List[1]+StrToFloat(SW_IBS_Ges_Euro_EK->Text)+StrToFloat(SS_Subangebot_Euro-
>Text)+
Bauteil1_Preis1[1]+Bauteil1_Preis2[1]+Bauteil1_Preis3[1]+Bauteil1_Preis4[1]+
Bauteil2_Preis1[1]+Bauteil2_Preis2[1]+Bauteil2_Preis3[1]+Bauteil2_Preis4[1]+
Bauteil3_Preis1[1]+Bauteil3_Preis2[1]+Bauteil3_Preis3[1]+Bauteil3_Preis4[1]+
Bauteil4_Preis1[1]+Bauteil4_Preis2[1]+Bauteil4_Preis3[1]+Bauteil4_Preis4[1];

Preis[1]= Preis[0];
Preis[2]= GesamtSumme_List[2];
Preis[3]= Preis[2];
AngebotZusammenstelKalkblattAnzeige("Summe Netto:",Preis,true);
RichEdit_Kalkulationsblatt->Lines->Add("");

if (KundenRabatt->Text != 0)
 {
 text="Rabatt "+KundenRabatt->Text+"%:";
 for(int                i=text.Length();i<=(103-FloatToStrF(GesamtSumme_List[2]/100*StrToFloat(KundenRabatt-
>Text),ffNumber,8,2).Length()-1);i++)
  {
  text=text+" ";
  }
 text=text   +"-"+   FloatToStrF(GesamtSumme_List[2]/100*StrToFloat(KundenRabatt->Text),ffNumber,8,2)+"
€";

 RichEdit_Kalkulationsblatt->SelAttributes->Style   =   RichEdit_Kalkulationsblatt->SelAttributes->Style   <<
fsBold;
 //RichEdit_Kalkulationsblatt->SelAttributes->Height -= 1;
 RichEdit_Kalkulationsblatt->Lines->Add(text);
 RichEdit_Kalkulationsblatt->Lines->Add("");

 Preis[0]=          Summe_List[1]+StrToFloat(SW_IBS_Ges_Euro_EK->Text)+StrToFloat(SS_Subangebot_Euro-
>Text)+
 Bauteil1_Preis1[1]+Bauteil1_Preis2[1]+Bauteil1_Preis3[1]+Bauteil1_Preis4[1]+
 Bauteil2_Preis1[1]+Bauteil2_Preis2[1]+Bauteil2_Preis3[1]+Bauteil2_Preis4[1]+
 Bauteil3_Preis1[1]+Bauteil3_Preis2[1]+Bauteil3_Preis3[1]+Bauteil3_Preis4[1]+
 Bauteil4_Preis1[1]+Bauteil4_Preis2[1]+Bauteil4_Preis3[1]+Bauteil4_Preis4[1];
 Preis[1]= Preis[0];
 Preis[2]= GesamtSumme_List[2]-GesamtSumme_List[2]/100*StrToFloat(KundenRabatt->Text);
 Preis[3]= Preis[2];
 AngebotZusammenstelKalkblattAnzeige("ENDSUMME Netto:",Preis,true);
 RichEdit_Kalkulationsblatt->Lines->Add("");
 }
}
//---------------------------------------------------------------------------
// Angebot erstellen inkl Kalkulation
//---------------------------------------------------------------------------
void __fastcall TF_Main::AngebotKalk(TStringGrid *StrGd, AnsiString text1, float Preis[4])
{
AnsiString text, textspeicher;
textspeicher=text1;
int h = 0, hh = 0;
float VKEP = 0, VKGP = 0;
for(int i=textspeicher.Length();i<=81;i++)
  {
  textspeicher=textspeicher+" ";
  }
if (RadioGroup_Preisauswahl->ItemIndex==0)
 {
```

```
 textspeicher=textspeicher+"Einzelpreis";
 }
for(int i=textspeicher.Length();i<=101;i++)
  {
  textspeicher=textspeicher+" ";
  }
textspeicher=textspeicher+"Gesamtpreis";


for (int s =1; s <= StrGd->RowCount; s++)
{
  if (StrGd->Cells[0][s] != "" )//&& StrGd->Cells[0][s]  == "OK")
  {
  text=StrGd->Cells[0][s]+" Stk.  "+StrGd->Cells[1][s];
  h = 1;
  if (RadioGroup_Preisauswahl->ItemIndex==0)
   {
   for(int i=text.Length();i<=(90-StrGd->Cells[15][s].Length());i++)
     {
     text=text+" ";
     }
   text=text+StrGd->Cells[15][s];
   for(int i=text.Length();i<=(110-StrGd->Cells[16][s].Length());i++)
     {
     text=text+" ";
     }
   text=text+StrGd->Cells[16][s];
   }
   if (h == 1 && hh == 0)
     {
     hh=1;
     RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style << fsBold;
     RichEdit_Gesamt->Lines->Add(textspeicher);
     RichEdit_Gesamt->Lines->Add("");
     }
  RichEdit_Gesamt->Lines->Add(text);
  Preis[0] += StrToFloat(StrGd->Cells[17][s]);
  Preis[1] += StrToFloat(StrGd->Cells[18][s]);
  Preis[2] += StrToFloat(StrGd->Cells[15][s]);
  Preis[3] += StrToFloat(StrGd->Cells[16][s]);
  VKEP += StrToFloat(StrGd->Cells[15][s]);
  VKGP += StrToFloat(StrGd->Cells[16][s]);
  RichEdit_Gesamt->Paragraph->Alignment = taLeftJustify;
  RichEdit_Gesamt->Lines->Add(StrGd->Cells[19][s]);
  RichEdit_Gesamt->Lines->Add(""); RichEdit_Gesamt->Lines->Add("");
  }
}
if ((RadioGroup_Preisauswahl->ItemIndex==0 || RadioGroup_Preisauswahl->ItemIndex==1) && VKGP !=0)
 {
 text="SUMME "+text1;
 for(int i=text.Length();i<=(110-FloatToStrF(VKGP,ffNumber,8,2).Length());i++)
   {
   text=text+" ";
   }
 text=text+FloatToStrF(VKGP,ffNumber,8,2)+" €";
 RichEdit_Gesamt->SelAttributes->Style = RichEdit_Gesamt->SelAttributes->Style << fsBold;
 RichEdit_Gesamt->Lines->Add(text);
 }
RichEdit_Gesamt->Lines->Add("");
RichEdit_Gesamt->Lines->Add("");
```

```
}
//---------------------------------------------------------------------------
// Kalkulationsblatt erstellen
//---------------------------------------------------------------------------
void __fastcall TF_Main::AngebotKalkblatt(TStringGrid *StrGd, AnsiString text1)
{
AnsiString text, textspeicher, text2;
textspeicher=text1;
int h = 0, hh = 0;
int anzahl = 0;
float VKEP = 0, VKGP = 0, EKEP = 0, EKGP = 0, DBges = 0, Proz =0;
float db = 0, dbproz = 0, help =0;

RichEdit_Kalkulationsblatt->Paragraph->Alignment = taLeftJustify;
for(int i=textspeicher.Length();i<=84;i++)
 {
 textspeicher=textspeicher+" ";
 }
textspeicher=textspeicher+"VKEP";
for(int i=textspeicher.Length();i<=99;i++)
  {
  textspeicher=textspeicher+" ";
  }
textspeicher=textspeicher+"VKGP";
for(int i=textspeicher.Length();i<=114;i++)
  {
  textspeicher=textspeicher+" ";
  }
textspeicher=textspeicher+"EKEP";
for(int i=textspeicher.Length();i<=129;i++)
  {
  textspeicher=textspeicher+" ";
  }
textspeicher=textspeicher+"EKGP";
for(int i=textspeicher.Length();i<=145;i++)
  {
  textspeicher=textspeicher+" ";
  }
textspeicher=textspeicher+"DB";
for(int i=textspeicher.Length();i<=160;i++)
  {
  textspeicher=textspeicher+" ";
  }
textspeicher=textspeicher+"%";

for (int s =1; s <= StrGd->RowCount; s++)
{
  if (StrGd->Cells[0][s] != "" )
  {
  h = 1;
  text = "";
  for(int i=text.Length();i<=(1-StrGd->Cells[0][s].Length());i++)
   {
   text=text+" ";
   }
  text2 = StrGd->Cells[19][s];
  text2 = text2.Delete(text2.Pos("\r\n"),text2.Length());
  text=text+StrGd->Cells[0][s]+" Stk.  "+StrGd->Cells[1][s] + " " + text2;
  if (text.Length() > 80)
     text = text.Delete(80,text.Length());
```

```
 for(int i=text.Length();i<=(88-StrGd->Cells[15][s].Length());i++)
  {
  text=text+" ";
  }
 text=text+StrGd->Cells[15][s];
 VKEP += StrToFloat(StrGd->Cells[15][s]);
 for(int i=text.Length();i<=(103-StrGd->Cells[16][s].Length());i++)
  {
  text=text+" ";
  }
 text=text+StrGd->Cells[16][s];
 VKGP += StrToFloat(StrGd->Cells[16][s]);
 for(int i=text.Length();i<=(118-StrGd->Cells[17][s].Length());i++)
  {
  text=text+" ";
  }
 text=text+StrGd->Cells[17][s];
 EKEP += StrToFloat(StrGd->Cells[17][s]);
 for(int i=text.Length();i<=(133-StrGd->Cells[18][s].Length());i++)
  {
  text=text+" ";
  }
 text=text+StrGd->Cells[18][s];
 EKGP += StrToFloat(StrGd->Cells[18][s]);
 db = StrToFloat(StrGd->Cells[16][s])-StrToFloat(StrGd->Cells[18][s]);
 for(int i=text.Length();i<=(148-FloatToStrF(db,ffNumber,8,2).Length());i++)
  {
  text=text+" ";
  }
 text=text+FloatToStrF(db,ffNumber,8,2);
 DBges += db;
 help = StrToFloat(StrGd->Cells[16][s]);
 if (help != 0)
 dbproz=(db / help)*100;
 else dbproz = 0;
for(int i=text.Length();i<=(163-FloatToStrF(dbproz,ffNumber,8,2).Length());i++)
  {
  text=text+" ";
  }
 text=text+FloatToStrF(dbproz,ffNumber,8,2);

 if (h == 1 && hh == 0)
  {
  hh=1;
  RichEdit_Kalkulationsblatt->SelAttributes->Style = RichEdit_Kalkulationsblatt->SelAttributes->Style <<
fsBold;
  //RichEdit_Kalkulationsblatt->SelAttributes->Height -= 1;
  RichEdit_Kalkulationsblatt->Lines->Add(textspeicher);
  RichEdit_Kalkulationsblatt->Lines->Add("");
  }
 //RichEdit_Kalkulationsblatt->SelAttributes->Height -= 1;
 RichEdit_Kalkulationsblatt->Lines->Add(text);

 int LastLine = RichEdit_Kalkulationsblatt->Lines->Count-1;
 int SelStart = 0;
 int SelLength = RichEdit_Kalkulationsblatt->Lines->Strings[LastLine].Length();
 //erstes Zeichen der letzten Linie suchen
 for(int i=0;i<LastLine;i++)
  {
  SelStart += RichEdit_Kalkulationsblatt->Lines->Strings[i].Length()+2;    //+2 wegen \r\n am Ende der Zeile
```

```
  }
  //Teile der letzte Zeile markieren
  RichEdit_Kalkulationsblatt->SelStart = SelStart+110;
  RichEdit_Kalkulationsblatt->SelLength = 25;
  RichEdit_Kalkulationsblatt->SelAttributes->Color = clRed;
  RichEdit_Kalkulationsblatt->SelStart = SelStart+140;
  RichEdit_Kalkulationsblatt->SelLength = 25;
  RichEdit_Kalkulationsblatt->SelAttributes->Color = clLime;

  //Cursor an letztes Zeichen setzen und Selectierung aufheben
  RichEdit_Kalkulationsblatt->SelStart = SelStart + SelLength+2;
  RichEdit_Kalkulationsblatt->SelLength = 0;
  //Text Farbe wieder auf schwarz setzen
  RichEdit_Kalkulationsblatt->SelAttributes->Color = clBlack;
  }
}
text="SUMME "+text1;
for(int i=text.Length();i<=(88-FloatToStrF(VKEP,ffNumber,8,2).Length());i++)
 {
 text=text+" ";
 }
text=text+FloatToStrF(VKEP,ffNumber,8,2)+" €";
for(int i=text.Length();i<=(103-FloatToStrF(VKGP,ffNumber,8,2).Length());i++)
 {
 text=text+" ";
 }
text=text+FloatToStrF(VKGP,ffNumber,8,2)+" €";
for(int i=text.Length();i<=(118-FloatToStrF(EKEP,ffNumber,8,2).Length());i++)
 {
 text=text+" ";
 }
text=text+FloatToStrF(EKEP,ffNumber,8,2)+" €";
for(int i=text.Length();i<=(133-FloatToStrF(EKGP,ffNumber,8,2).Length());i++)
 {
 text=text+" ";
 }
text=text+FloatToStrF(EKGP,ffNumber,8,2)+" €";
for(int i=text.Length();i<=(148-FloatToStrF(DBges,ffNumber,8,2).Length());i++)
 {
 text=text+" ";
 }
text=text+FloatToStrF(DBges,ffNumber,8,2)+" €";
if (VKGP != 0)
   Proz=(DBges / VKGP)*100;
else Proz = 0;
for(int i=text.Length();i<=(163-FloatToStrF(Proz,ffNumber,8,2).Length());i++)
 {
 text=text+" ";
 }
text=text+FloatToStrF(Proz,ffNumber,8,2)+" %";

RichEdit_Kalkulationsblatt->SelAttributes->Style = RichEdit_Kalkulationsblatt->SelAttributes->Style <<
fsBold;
//RichEdit_Kalkulationsblatt->SelAttributes->Height -= 1;
RichEdit_Kalkulationsblatt->Lines->Add(text);

int LastLine = RichEdit_Kalkulationsblatt->Lines->Count-1;
int SelStart = 0;
int SelLength = RichEdit_Kalkulationsblatt->Lines->Strings[LastLine].Length();
//erstes Zeichen der letzten Linie suchen
```

```
for(int i=0;i<LastLine;i++)
 {
 SelStart += RichEdit_Kalkulationsblatt->Lines->Strings[i].Length()+2;    //+2 wegen \r\n am Ende der Zeile
 }
//Teile der letzte Zeile markieren
RichEdit_Kalkulationsblatt->SelStart = SelStart+110;
RichEdit_Kalkulationsblatt->SelLength = 27;
RichEdit_Kalkulationsblatt->SelAttributes->Color = clRed;
RichEdit_Kalkulationsblatt->SelStart = SelStart+140;
RichEdit_Kalkulationsblatt->SelLength = 27;
RichEdit_Kalkulationsblatt->SelAttributes->Color = clLime;
//Cursor an letztes Zeichen setzen und Selectierung aufheben
RichEdit_Kalkulationsblatt->SelStart = SelStart + SelLength+2;
RichEdit_Kalkulationsblatt->SelLength = 0;
//Text Farbe wieder auf schwarz setzen
RichEdit_Kalkulationsblatt->SelAttributes->Color = clBlack;

RichEdit_Kalkulationsblatt->Lines->Add("");
}
//---------------------------------------------------------------------------
// Angebot Zusammenstellung einzelne Bauteile erstellen
//---------------------------------------------------------------------------
void __fastcall TF_Main::AngebotZusammenstel(float I, float II, float III, float IV)
{
AnsiString text="";
if (I != 0)
 {
 text="I.   Fühler und Geber:";
 for(int i=text.Length();i<=(90-FloatToStrF(I,ffNumber,8,2).Length());i++)
  {
  text=text+" ";
  }
 text=text + FloatToStrF(I,ffNumber,8,2);
 RichEdit_Gesamt->Lines->Add(text);
 }
if (II != 0)
 {
 text="II.  DDC:";
 for(int i=text.Length();i<=(90-FloatToStrF(II,ffNumber,8,2).Length());i++)
  {
  text=text+" ";
  }
 text=text + FloatToStrF(II,ffNumber,8,2);
 RichEdit_Gesamt->Lines->Add(text);
 }
if (III != 0 && SubangebotSS->Checked == false)
 {
 text="III. Schaltschrank:";
 for(int i=text.Length();i<=(90-FloatToStrF(III,ffNumber,8,2).Length());i++)
  {
  text=text+" ";
  }
 text=text + FloatToStrF(III,ffNumber,8,2);
 RichEdit_Gesamt->Lines->Add(text);
 }
if (IV != 0)
 {
 text="IV.  Schaltschrank Beistellgeräte:";
 for(int i=text.Length();i<=(90-FloatToStrF(IV,ffNumber,8,2).Length());i++)
  {
```

```
 text=text+" ";
 }
text=text + FloatToStrF(IV,ffNumber,8,2);
RichEdit_Gesamt->Lines->Add(text);
 }
}
//---------------------------------------------------------------------------
// Angebot Zusammenstellung einzelne Bauteile Kalkulationsblatt erstellen
//---------------------------------------------------------------------------
void __fastcall TF_Main::AngebotZusammenstelKalkblatt(AnsiString text, float PreisI[4], float PreisII[4], float
PreisIII[4], float PreisIV[4])
{
float Preis[4];
AngebotZusammenstelKalkblattAnzeige("I.   Fühler und Geber:",PreisI,false);
AngebotZusammenstelKalkblattAnzeige("II.  DDC:",PreisII,false);
if (SubangebotSS->Checked == false)
  AngebotZusammenstelKalkblattAnzeige("III. Schaltschrank:",PreisIII,false);
AngebotZusammenstelKalkblattAnzeige("IV.  Schaltschrank Beistellgeräte:",PreisIV,false);
Preis[0] = PreisI[0]+PreisII[0]+PreisIII[0]+PreisIV[0];
Preis[1] = PreisI[1]+PreisII[1]+PreisIII[1]+PreisIV[1];
Preis[2] = PreisI[2]+PreisII[2]+PreisIII[2]+PreisIV[2];
Preis[3] = PreisI[3]+PreisII[3]+PreisIII[3]+PreisIV[3];
AngebotZusammenstelKalkblattAnzeige("Gesamtkosten "+text+":",Preis,true);
RichEdit_Kalkulationsblatt->Lines->Add("");

}
//---------------------------------------------------------------------------
// Angebot Zusammenstellung einzelne Bauteile Kalkulationsblatt ANZEIGE erstellen
//---------------------------------------------------------------------------
void __fastcall TF_Main::AngebotZusammenstelKalkblattAnzeige(AnsiString text, float Preis[4],bool fett)
{
float db=0, Proz=0;
for(int i=text.Length();i<=(88-FloatToStrF(Preis[2],ffNumber,8,2).Length());i++)
 {
 text=text+" ";
 }
text=text + FloatToStrF(Preis[2],ffNumber,8,2) + " €";
for(int i=text.Length();i<=(103-FloatToStrF(Preis[3],ffNumber,8,2).Length());i++)
 {
 text=text+" ";
 }
text=text + FloatToStrF(Preis[3],ffNumber,8,2) + " €";
for(int i=text.Length();i<=(118-FloatToStrF(Preis[0],ffNumber,8,2).Length());i++)
 {
 text=text+" ";
 }
text=text + FloatToStrF(Preis[0],ffNumber,8,2) + " €";
for(int i=text.Length();i<=(133-FloatToStrF(Preis[1],ffNumber,8,2).Length());i++)
 {
 text=text+" ";
 }
text=text + FloatToStrF(Preis[1],ffNumber,8,2) + " €";
db = Preis[3]-Preis[1];
for(int i=text.Length();i<=(148-FloatToStrF(db,ffNumber,8,2).Length());i++)
 {
 text=text+" ";
 }
text=text + FloatToStrF(db,ffNumber,8,2) + " €";
if (Preis[3] != 0)
   Proz=(db / Preis[3])*100;
```

```
else Proz = 0;
for(int i=text.Length();i<=(163-FloatToStrF(Proz,ffNumber,8,2).Length());i++)
 {
 text=text+" ";
 }
text=text + FloatToStrF(Proz,ffNumber,8,2) + " %";

if (fett) RichEdit_Kalkulationsblatt->SelAttributes->Style = RichEdit_Kalkulationsblatt->SelAttributes->Style
<< fsBold;
//RichEdit_Kalkulationsblatt->SelAttributes->Height -= 1;
RichEdit_Kalkulationsblatt->Lines->Add(text);

int LastLine = RichEdit_Kalkulationsblatt->Lines->Count-1;
int SelStart = 0;
int SelLength = RichEdit_Kalkulationsblatt->Lines->Strings[LastLine].Length();
//erstes Zeichen der letzten Linie suchen
for(int i=0;i<LastLine;i++)
 {
 SelStart += RichEdit_Kalkulationsblatt->Lines->Strings[i].Length()+2;    //+2 wegen \r\n am Ende der Zeile
 }
//Teile der letzte Zeile markieren
RichEdit_Kalkulationsblatt->SelStart = SelStart+110;
RichEdit_Kalkulationsblatt->SelLength = 27;
RichEdit_Kalkulationsblatt->SelAttributes->Color = clRed;
RichEdit_Kalkulationsblatt->SelStart = SelStart+140;
RichEdit_Kalkulationsblatt->SelLength = 27;
RichEdit_Kalkulationsblatt->SelAttributes->Color = clLime;
//Cursor an letztes Zeichen setzen und Selectierung aufheben
RichEdit_Kalkulationsblatt->SelStart = SelStart + SelLength+2;
RichEdit_Kalkulationsblatt->SelLength = 0;
//Text Farbe wieder auf schwarz setzen
RichEdit_Kalkulationsblatt->SelAttributes->Color = clBlack;

}
//---------------------------------------------------------------------------
// LV ins Excel exportieren ...
//---------------------------------------------------------------------------

void __fastcall TF_Main::Button_ExportLVClick(TObject *Sender)
{

TProgressBar *p = new TProgressBar(StatusBar1);
p->Parent = StatusBar1;
p->Top = 2;
p->Left = StatusBar1->Panels->Items[0]->Width + 2;
p->Width = StatusBar1->Panels->Items[1]->Width -2;
p->Height = StatusBar1->Height-2;
p->Smooth = true;
p->Step = 1;
p->Min = 0;
//p->Max = 11;
p->Max = AnzahlZeilenExpImp*10;
StatusBar1->Repaint();

Table1->Active = false;

Variant Excel;

try
 {
```

```
 Excel = GetActiveOleObject("Excel.Application");
 }
catch(...)
 {
 Excel = CreateOleObject("Excel.Application");
 }
//  Excel.OlePropertySet("Visible", true);

try
 {
 Variant WorkBooks = Excel.OlePropertyGet("WorkBooks");
 //WorkBooks.OleFunction("Add");
 WorkBooks.OleFunction("Open",Dir + "\\files\\Exp_LV_orig.xls");
 }
catch(...)
 {
 }
p->StepIt();
Variant ActiveWorkBook = Excel.OlePropertyGet("ActiveWorkbook");

Variant WorkSheets = Excel.OlePropertyGet("Worksheets");
int i;
for (i=1;i<=10;i++)
 {
 Variant WorkSheet = WorkSheets.OlePropertyGet("Item", i);
 WorkSheet.OleFunction("Activate");

 int iRow, iCol;
 try
  {
  //Beschreiben der Excelliste
  for (iRow=1; iRow <= AnzahlZeilenExpImp; iRow++)
   {
    //Fortschrittsanzeige +1 setzen
    p->StepIt();
    for (iCol=1; iCol <=11; iCol++)
     {
      Variant Range = WorkSheet.OlePropertyGet("Cells", iRow+1, iCol);
      switch(i)
      {
      case 1 : Range.OlePropertySet("Value", pStringGrid_Baut1->Cells[iCol][iRow]); break;
      case 2 : Range.OlePropertySet("Value", pStringGrid_Baut2->Cells[iCol][iRow]); break;
      case 3 : Range.OlePropertySet("Value", pStringGrid_Baut3->Cells[iCol][iRow]); break;
      case 4 : Range.OlePropertySet("Value", pStringGrid_Baut4->Cells[iCol][iRow]); break;
      case 5 : Range.OlePropertySet("Value", pStringGrid_Baut5->Cells[iCol][iRow]); break;
      case 6 : Range.OlePropertySet("Value", pStringGrid_Baut6->Cells[iCol][iRow]); break;
      case 7 : Range.OlePropertySet("Value", pStringGrid_Baut7->Cells[iCol][iRow]); break;
      case 8 : Range.OlePropertySet("Value", pStringGrid_Baut8->Cells[iCol][iRow]); break;
      case 9 : Range.OlePropertySet("Value", pStringGrid_Baut9->Cells[iCol][iRow]); break;
      case 10 : Range.OlePropertySet("Value", pStringGrid_Baut10->Cells[iCol][iRow]); break;
      };
     }
   }
  //ShowMessage("Daten wurden in Excel exportiert!");
  }
 catch(...)
  {
  ShowMessage("Fehler bei Datenexport! in Registerblatt: "+IntToStr(i)+" in Zeile: "+IntToStr(iRow)+" Spalte:
"+ IntToStr(iCol));
  }
```

```
 }
try
 {
 ActiveWorkBook.OleFunction("SaveAs", Dir + "\\Exp_LV.xls");
 }
catch(...)
 {
 }
Excel.OleFunction("Quit");
Excel = Unassigned;

Table1->Active = true;

//Fortschrittsanzeige auf 0 setzen
p->Position = 0;
delete p;
}
//---------------------------------------------------------------------------
// LV aus Excel importieren ...
//---------------------------------------------------------------------------

void __fastcall TF_Main::Button_ImportLVClick(TObject *Sender)
{

TProgressBar *p = new TProgressBar(StatusBar1);
p->Parent = StatusBar1;
p->Top = 2;
p->Left = StatusBar1->Panels->Items[0]->Width + 2;
p->Width = StatusBar1->Panels->Items[1]->Width -2;
p->Height = StatusBar1->Height-2;
p->Smooth = true;
p->Step = 1;
p->Min = 0;
//p->Max = 11;
p->Max = AnzahlZeilenExpImp*10;
StatusBar1->Repaint();

Table1->Active = false;

Variant Excel;

try
 {
  Excel = GetActiveOleObject("Excel.Application");
 }
catch(...)
 {
  Excel = CreateOleObject("Excel.Application");
 }
//  Excel.OlePropertySet("Visible", true);

try
 {
  Variant WorkBooks = Excel.OlePropertyGet("WorkBooks");
  //WorkBooks.OleFunction("Add");
  WorkBooks.OleFunction("Open",Dir + "\\Exp_LV.xls");
 }
catch(...)
 {
 }
```

```
Variant ActiveWorkBook = Excel.OlePropertyGet("ActiveWorkbook");
Variant WorkSheets = Excel.OlePropertyGet("Worksheets");
int i;
for (i=1;i<=10;i++)
 {
  Variant WorkSheet = WorkSheets.OlePropertyGet("Item", i);
  WorkSheet.OleFunction("Activate");

  int iRow, iCol;
  try
   {
    //Auslesen der Excelliste - Einlesen in die Datenbank
    for (iRow=1; iRow <= AnzahlZeilenExpImp; iRow++)
     {
      //Fortschrittsanzeige +1 setzen
      p->StepIt();
      //Daten nun Importieren
      for (iCol=1; iCol <=11; iCol++)
       {
        AnsiString Wert = WorkSheet.OlePropertyGet("Cells",iRow+1, iCol).OlePropertyGet("Value");
        switch(i)
          {
          case 1 : pStringGrid_Baut1->Cells[iCol][iRow] = Wert; break;
          case 2 : pStringGrid_Baut2->Cells[iCol][iRow] = Wert; break;
          case 3 : pStringGrid_Baut3->Cells[iCol][iRow] = Wert; break;
          case 4 : pStringGrid_Baut4->Cells[iCol][iRow] = Wert; break;
          case 5 : pStringGrid_Baut5->Cells[iCol][iRow] = Wert; break;
          case 6 : pStringGrid_Baut6->Cells[iCol][iRow] = Wert; break;
          case 7 : pStringGrid_Baut7->Cells[iCol][iRow] = Wert; break;
          case 8 : pStringGrid_Baut8->Cells[iCol][iRow] = Wert; break;
          case 9 : pStringGrid_Baut9->Cells[iCol][iRow] = Wert; break;
          case 10 : pStringGrid_Baut10->Cells[iCol][iRow] = Wert; break;
          };
       }
      switch (i)
       {
       case  1 : CheckForDevice(iRow,pStringGrid_Baut1);
                 break;
       case  2 : CheckForDevice(iRow,pStringGrid_Baut2);
                 break;
       case  3 : CheckForDevice(iRow,pStringGrid_Baut3);
                 break;
       case  4 : CheckForDevice(iRow,pStringGrid_Baut4);
                 break;
       case  5 : CheckForDevice(iRow,pStringGrid_Baut5);
                 break;
       case  6 : CheckForDevice(iRow,pStringGrid_Baut6);
                 break;
       case  7 : CheckForDevice(iRow,pStringGrid_Baut7);
                 break;
       case  8 : CheckForDevice(iRow,pStringGrid_Baut8);
                 break;
       case  9 : CheckForDevice(iRow,pStringGrid_Baut9);
                 break;
       case  10: CheckForDevice(iRow,pStringGrid_Baut10);
                 break;
       };
     }
  }
```

```
  catch(...)
   {
   ShowMessage("Fehler bei Datenimport! in Registerblatt: "+IntToStr(i)+" in Zeile: "+IntToStr(iRow)+" Spalte:
"+ IntToStr(iCol));
   }
 }
Excel.OleFunction("Quit");
Excel = Unassigned;

Table1->Active = true;

//Holen der Information aus einem der 10 verdeckten StringGrids in das sichtbare
switch (RadioGroup1->ItemIndex)
{
  case 0 :
    // Bauteil 1 ausgewählt
    for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
     {
     for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
      {
       StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut1->Cells[iCol][iRow];
      }
     }
    break;
  case 1 :
   //  if (pStringGrid_Baut2->Cells[0][0] == "") break;
    // Bauteil 2 ausgewählt
    for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
     {
     for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
      {
       StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut2->Cells[iCol][iRow];
      }
     }
    break;
  case 2 :
     if (pStringGrid_Baut3->Cells[0][0] == "") break;
    // Bauteil 3 ausgewählt
    for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
     {
     for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
      {
       StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut3->Cells[iCol][iRow];
      }
     }
    break;
  case 3 :
     if (pStringGrid_Baut4->Cells[0][0] == "") break;
    // Bauteil 4 ausgewählt
    for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
     {
     for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
      {
       StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut4->Cells[iCol][iRow];
      }
     }
    break;
  case 4 :
    // Bauteil 5 ausgewählt
     if (pStringGrid_Baut5->Cells[0][0] == "") break;
```

```
  for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
   {
   for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
    {
     StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut5->Cells[iCol][iRow];
    }
   }
  break;
 case 5 :
   if (pStringGrid_Baut6->Cells[0][0] == "") break;
  // Bauteil 6 ausgewählt
  for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
   {
   for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
    {
     StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut6->Cells[iCol][iRow];
    }
   }
  break;
 case 6 :
   if (pStringGrid_Baut7->Cells[0][0] == "") break;
  // Bauteil 7 ausgewählt
  for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
   {
   for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
    {
     StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut7->Cells[iCol][iRow];
    }
   }
  break;
 case 7 :
   if (pStringGrid_Baut8->Cells[0][0] == "") break;
  // Bauteil 8 ausgewählt
  for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
   {
   for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
    {
     StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut8->Cells[iCol][iRow];
    }
   }
  break;
 case 8 :
   if (pStringGrid_Baut9->Cells[0][0] == "") break;
  // Bauteil 9 ausgewählt
  for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
   {
   for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
    {
     StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut9->Cells[iCol][iRow];
    }
   }
  break;
 case 9 :
   if (pStringGrid_Baut10->Cells[0][0] == "") break;
  // Bauteil 10 ausgewählt
  for (int iRow = 0; iRow <= StringGrid1->RowCount; iRow++)
   {
   for (int iCol = 0; iCol <= StringGrid1->ColCount; iCol++)
    {
     StringGrid1->Cells[iCol][iRow] = pStringGrid_Baut10->Cells[iCol][iRow];
```

```
     }
    }
   break;
}
//Fortschrittsanzeige auf 0 setzen
p->Position = 0;
delete p;
}
//---------------------------------------------------------------------------
// Daten Exp aus DB2 ...
//---------------------------------------------------------------------------
void __fastcall TF_Main::Button_Exp_DB2Click(TObject *Sender)
{

Table2->Active = false;
Variant Excel;
try
 {
  Excel = GetActiveOleObject("Excel.Application");
 }
catch(...)
 {
  Excel = CreateOleObject("Excel.Application");
 }
//  Excel.OlePropertySet("Visible", true);

try
 {
  Variant WorkBooks = Excel.OlePropertyGet("WorkBooks");
  //WorkBooks.OleFunction("Add");
  WorkBooks.OleFunction("Open",Dir + "\\files\\Exp_Datenbank2_orig.xls");
 }
catch(...)
 {
 }

Variant ActiveWorkBook = Excel.OlePropertyGet("ActiveWorkbook");

Variant WorkSheets = Excel.OlePropertyGet("Worksheets");

Variant WorkSheet = WorkSheets.OlePropertyGet("Item", 1);
WorkSheet.OleFunction("Activate");

Table2->Open();
Query2->Close();
Query2->SQL->Clear();
AnsiString ASSQLCommand = "SELECT * FROM Angebot_DB2_Aktuell ORDER BY ID";
Query2->SQL->Add(ASSQLCommand);
Query2->Open();
Query2->Active = true;
Query2->First();

//MaxWert für Fortschrittsanzeige setzen
ProgressBar_DB2->Max = Query2->RecordCount;

int iRow, iCol;
try
 {
  //Beschreiben der Excelliste
  for (iRow=2; iRow <= AnzahlZeilenExpImp; iRow++)
```

```
  {
    //Fortschrittsanzeige +1 setzen
    ProgressBar_DB2->StepIt();
    for (iCol=1; iCol <=19; iCol++)
      {
       Variant Range = WorkSheet.OlePropertyGet("Cells", iRow, iCol);
       Range.OlePropertySet("Value", Query2->Fields->Fields[iCol-1]->AsString);
      }
    Query2->Next();
    //Bei Eof Abbruch des Exports
    if (Query2->Eof) break;

  }
 //ShowMessage("Daten wurden in Excel exportiert!");
 }
catch(...)
 {
  ShowMessage("Fehler bei Datenexport! in Zeile: "+IntToStr(iRow)+" Spalte: "+ IntToStr(iCol));
 }

try
 {
 ActiveWorkBook.OleFunction("SaveAs", Dir + "\\Exp_Datenbank2.xls");
 }
catch(...)
 {
 }
Excel.OleFunction("Quit");
Excel = Unassigned;

Table2->Active = true;

//Fortschrittsanzeige auf 0 setzen
ProgressBar_DB2->Position = 0;
}
//---------------------------------------------------------------------------
// ExcelListe in DB2 importieren
//---------------------------------------------------------------------------
void __fastcall TF_Main::Button_Imp_DB2Click(TObject *Sender)
{

Table2->Active = false;

Variant Excel;

try
 {
  Excel = GetActiveOleObject("Excel.Application");
 // Excel = CreateOleObject("Excel.Application");
 }
catch(...)
 {
  Excel = CreateOleObject("Excel.Application");
 }
//  Excel.OlePropertySet("Visible", true);

try
 {
  Variant WorkBooks = Excel.OlePropertyGet("WorkBooks");
  //WorkBooks.OleFunction("Add");
```

```
  WorkBooks.OleFunction("Open",Dir + "\\Exp_Datenbank2.xls");
 }
catch(...)
 {
 }

Variant ActiveWorkBook = Excel.OlePropertyGet("ActiveWorkbook");

Variant WorkSheets = Excel.OlePropertyGet("Worksheets");

Variant WorkSheet = WorkSheets.OlePropertyGet("Item", 1);
WorkSheet.OleFunction("Activate");

int iRow, iCol;
int ptr = 0, ptr1 = 0;
try
 {
  //Auslesen der Excelliste - bis Grenze wg Statusanzeige
  for (iRow=2; iRow <= AnzahlZeilenExpImp; iRow++)
   {
    //Überprüfen letzen Eintrag in Excel
    ptr = AnsiCompareStr (WorkSheet.OlePropertyGet("Cells",iRow, 1).OlePropertyGet("Value"),"");
    if ( ptr == 0)
       break;
   }
   //MaxWert für Fortschrittsanzeige setzen
   ProgressBar_DB2->Max = iRow
   ;
  //Auslesen der Excelliste - Einlesen in die Datenbank
  for (iRow=2; iRow <= AnzahlZeilenExpImp; iRow++)
   {
    //Fortschrittsanzeige +1 setzen
    ProgressBar_DB2->StepIt();
    //Überprüfen letzen Eintrag in Excel
    ptr = AnsiCompareStr (WorkSheet.OlePropertyGet("Cells",iRow, 1).OlePropertyGet("Value"),"");
    if ( ptr == 0)
       break;
    //ist eine ID nicht vorhanden - neuen Datensatz mit dieser ID erstellen
    Table2->Open();
    Query2->Close();
    Query2->SQL->Clear();
    AnsiString ASSQLCommand = "SELECT * FROM Angebot_DB2_Aktuell ORDER BY ID";
    Query2->SQL->Add(ASSQLCommand);
    Query2->Open();
    Query2->Active = true;
    Query2->First();
    while (!Query2->Eof)
     {
       ptr1 = AnsiCompareStr (WorkSheet.OlePropertyGet("Cells",iRow, 1).OlePropertyGet("Value"),Query2-
>FieldByName("ID")->AsString);
       if (ptr1 == 0)
          break;
       Query2->Next();
     }
    if (ptr1 != 0)
       {
       Table2->Open();
       Query2->Close();
       Query2->SQL->Clear();       //SQL Commando insert Angebot_DB_Aktuell mit der Spalte "ID" mit Wert
aus der "aktiven" Celle  wo Id der Zeile
```

```
      AnsiString ASSQLCommand = "Insert Into Angebot_DB2_Aktuell (ID) VALUES ("+
WorkSheet.OlePropertyGet("Cells",iRow, 1).OlePropertyGet("Value") +")";
      Query2->SQL->Add(ASSQLCommand);
      Query2->ExecSQL();
       }
    //Daten nun Importieren
    for (iCol=1; iCol <=19; iCol++)
     {

      AnsiString AStemp = "";
      switch (iCol)   //In welcher Spalte man sich grad befindet
      {                     //Zuweisung der Spaltenbezeichnung
       case 1: AStemp ="ID";
             break;
       case 2: AStemp ="Bezeichnung";
             break;
       case 3: AStemp ="Gerätetyp";
             break;
       case 4: AStemp ="Gerätetyp2";
             break;
       case 5: AStemp ="Gerätetyp3";
             break;
       case 6: AStemp ="AE";
             break;
       case 7: AStemp ="AA";
             break;
       case 8: AStemp ="DE";
             break;
       case 9: AStemp ="DA";
             break;
       case 10: AStemp ="Hersteller";
             break;
       case 11: AStemp ="RabattgruppeEK";
             break;
       case 12: AStemp ="RabattgruppeVK";
             break;
       case 13: AStemp ="ListenpreisType1";
             break;
       case 14: AStemp ="ListenpreisType2";
             break;
       case 15: AStemp ="ListenpreisType3";
             break;
       case 16: AStemp ="BMGruppe";
             break;
       case 17: AStemp ="DatenpktXLWeb";
             break;
       case 18: AStemp ="Datenpunkte";
             break;
       case 19: AStemp ="Ausschreibungstext";
             break;
       default: return;
      }
      AnsiString Wert = WorkSheet.OlePropertyGet("Cells",iRow, iCol).OlePropertyGet("Value");
      int i = 0;
      i = Wert.Pos(',');
      // In der Preisspalte muss das Komma durch einen Punkt beim Import ersetzt werden
      if (i != 0 && (iCol == 13 || iCol == 14 || iCol == 15))
       {
        Wert.Delete(i,1);
        Wert.Insert('.',i);
```

```
        }
      //Bei Spalte Datenpunkte(Wahr oder Falsch) wird das schon interpretiert - für DB aber True oder False notwendig!!!
      if ((AnsiCompareStr(Wert, "-1") == 0) && (iCol == 18))
         Wert = "True";
      if ((AnsiCompareStr(Wert, "0") == 0) && (iCol == 18))
         Wert = "False";
      Table2->Open();
      Query2->Close();
      Query2->SQL->Clear();      //SQL Commando update Angebot_DB_Aktuell mit der Spalte "AStemp" mit Wert aus der "aktiven" Celle  wo Id der Zeile suchen nach Komma und durch punkt ersetzen: AnsiString.pos while AnsiString.pos"/delete/insert," durch "."
      AnsiString ASSQLCommand = "Update Angebot_DB2_Aktuell SET " + AStemp + "='" + Wert + "' WHERE ID = '" + WorkSheet.OlePropertyGet("Cells",iRow, 1).OlePropertyGet("Value") +"'";
      Query2->SQL->Add(ASSQLCommand);
      Query2->ExecSQL();
     }
   }
  //ShowMessage("Daten wurden in Excel exportiert!");
 }
catch(...)
 {
  ShowMessage("Fehler bei Datenimport! in Zeile: "+IntToStr(iRow)+" Spalte: "+ IntToStr(iCol));
 }

Excel.OleFunction("Quit");
Excel = Unassigned;

Table2->Active = true;

//Fortschrittsanzeige auf 0 setzen
ProgressBar_DB2->Position = 0;
}
//---------------------------------------------------------------------------
// Speedbutton "Save"
//---------------------------------------------------------------------------
void __fastcall TF_Main::SpeedButton_SaveClick(TObject *Sender)
{
try
{
Schreiben(StatusBar1->Panels->Items[1]->Text);
F_Main->Caption = "DI_Arbeit   " + StatusBar1->Panels->Items[1]->Text;
AenderungDurchgefuhert = false;
}
catch (...)
{
SaveDialog1->Filter="Projekt-Datein *.prjl*.prj";
if (SaveDialog1->Execute())
   {
  //Memo1->Lines->SaveToFile(SaveDialog1->FileName);
     Schreiben(SaveDialog1->FileName);
     StatusBar1->Panels->Items[1]->Text = SaveDialog1->FileName;
     F_Main->Caption = "DI_Arbeit   " + SaveDialog1->FileName;
     AenderungDurchgefuhert = false;
   }
}
}
//---------------------------------------------------------------------------
// Speedbutton "Öffnen"
//---------------------------------------------------------------------------
```

```
void __fastcall TF_Main::SpeedButton_OpenClick(TObject *Sender)
{
OpenDialog1->Filter="Projekt-Datein *.prj|*.prj";
//OpenDialog1->Execute();    Wird in der If Abfrage aufgerufen
if (OpenDialog1->Execute())
  // Memo1->Lines->Add(OpenDialog1->FileName); Dateinamen einfügen
   // Memo1->Lines->LoadFromFile(OpenDialog1->FileName);
   {   //Reset der eingegebenen Bauteile
       Reset1Click(Sender);
       StatusBar1->Panels->Items[1]->Text = OpenDialog1->FileName;
       F_Main->Caption = "DI_Arbeit   " + OpenDialog1->FileName;
       Lesen(OpenDialog1->FileName);
       // Nach öffnen in der Eingabemaske auf default Bauteil1 und laden von Bauteil1 in StringGrid1
       RadioButtonBautSav = -1;
       RadioGroup1->ItemIndex = 0;
       AenderungDurchgefuhert = false;
       //eingegebenen Bauteile übernehmen ? Sollte noch eine abfrage kommen
      if (OKBottomDlg->ShowModal() == mrOk) Button7Click(Sender);
   }
}
//---------------------------------------------------------------------------
// Speedbutton "Kalkulation Übersicht"
//---------------------------------------------------------------------------

void __fastcall TF_Main::SpeedButton_KClick(TObject *Sender)
{
  PageControl1->Visible = false;

  Panel4->Visible = false;

  Panel5->Visible = false;

  Panel1->Visible = false;
  pComboEditor->Visible = false;

  Panel3->Visible = true;

  Panel2->Visible = false;
  Table1->Active = false;

  Panel6->Visible = false;
  Table2->Active = false;

  OptionenPanel->Visible = false;
}
//---------------------------------------------------------------------------
// Speedbutton "LV Eingabe"
//---------------------------------------------------------------------------

void __fastcall TF_Main::SpeedButton_LVClick(TObject *Sender)
{
  PageControl1->Visible = false;

  Panel4->Visible = false;

  Panel5->Visible = false;

  Panel1->Visible = true;

  Panel3->Visible = false;
```

```
  Panel2->Visible = false;
  Table1->Active = false;

  Panel6->Visible = false;
  Table2->Active = false;

  OptionenPanel->Visible = false;
}
//---------------------------------------------------------------------------
// Eingabe der DDC für alle Bauteile gleichzeitig - bei neuer Datenübernahme aber DDC gelöscht!!!
//---------------------------------------------------------------------------
void __fastcall TF_Main::ComboBox_DDC_OptionChange(TObject *Sender)
{
pKalkulation_Baut1->ComboBox_DDC->ItemIndex = ComboBox_DDC_Option->ItemIndex;
pKalkulation_Baut2->ComboBox_DDC->ItemIndex = ComboBox_DDC_Option->ItemIndex;
pKalkulation_Baut3->ComboBox_DDC->ItemIndex = ComboBox_DDC_Option->ItemIndex;
pKalkulation_Baut4->ComboBox_DDC->ItemIndex = ComboBox_DDC_Option->ItemIndex;
pKalkulation_Baut5->ComboBox_DDC->ItemIndex = ComboBox_DDC_Option->ItemIndex;
pKalkulation_Baut6->ComboBox_DDC->ItemIndex = ComboBox_DDC_Option->ItemIndex;
pKalkulation_Baut7->ComboBox_DDC->ItemIndex = ComboBox_DDC_Option->ItemIndex;
pKalkulation_Baut8->ComboBox_DDC->ItemIndex = ComboBox_DDC_Option->ItemIndex;
pKalkulation_Baut9->ComboBox_DDC->ItemIndex = ComboBox_DDC_Option->ItemIndex;
pKalkulation_Baut10->ComboBox_DDC->ItemIndex = ComboBox_DDC_Option->ItemIndex;
}
//---------------------------------------------------------------------------
// Berechnung der Prozente über Rabattblatt Verkauf für Kalkulation
//---------------------------------------------------------------------------

float __fastcall TF_Main::Rabkalk(AnsiString RabGr)
{
float Prozent = 1;
int comp=0;
for (int i=0;i<=StringGrid_VK->RowCount;i++)
 {
 comp = AnsiCompareStr(RabGr, StringGrid_VK->Cells[0][i]);
 if (comp == 0)
   {
    Prozent = 1-StrToFloat(StringGrid_VK->Cells[1][i])/100;
    break;
   }
 }
return (Prozent);
}
//---------------------------------------------------------------------------
// Berechnung der Prozente über Rabattblatt Einkauf für Kalkulation
//---------------------------------------------------------------------------

float __fastcall TF_Main::RabkalkEK(AnsiString RabGr)
{
float Prozent = 1;
int comp=0;
for (int i=0;i<=StringGrid_EK->RowCount;i++)
 {
 comp = AnsiCompareStr(RabGr, StringGrid_EK->Cells[0][i]);
 if (comp == 0)
   {
    Prozent = 1-StrToFloat(StringGrid_EK->Cells[1][i])/100;
    break;
   }
```

```
 }
return (Prozent);
}
//---------------------------------------------------------------------------
// SS Subangebot - alle anderen SS Eingaben ausblenden
//---------------------------------------------------------------------------

void __fastcall TF_Main::SubangebotSSClick(TObject *Sender)
{
if (SubangebotSS->Checked)
 {
 SS_Subangebot_Euro->Enabled = true;
 SS_Aufschlag->Enabled = true;
 UpDown2->Enabled = true;
 Label86->Enabled = true;
 Label87->Enabled = true;
 }
 else
 {
 SS_Subangebot_Euro->Enabled = false;
 SS_Subangebot_Euro->Text = "0,00";
 SS_Aufschlag->Enabled = false;
 UpDown2->Enabled = false;
 Label86->Enabled = false;
 Label87->Enabled = false;
 }

}
//---------------------------------------------------------------------------
void __fastcall TF_Main::SS_Subangebot_EuroChange(TObject *Sender)
{
try
 {
 SS_Subangebot_Euro->Text = FloatToStrF(StrToFloat(SS_Subangebot_Euro->Text),ffFixed,8,2);
 History_SS_Subangebot_Euro = SS_Subangebot_Euro->Text;
 }
catch(...)
 {
 ShowMessage("Bitte geben sie einen Zahlenwert ein!!!");
 SS_Subangebot_Euro->Text = History_SS_Subangebot_Euro;
 }
}
//---------------------------------------------------------------------------
// Export Angebot in Excel
//---------------------------------------------------------------------------
void __fastcall TF_Main::Export_GesamtClick(TObject *Sender)
{
TProgressBar *p = new TProgressBar(StatusBar1);
p->Parent = StatusBar1;
p->Top = 2;
p->Left = StatusBar1->Panels->Items[0]->Width + 2;
p->Width = StatusBar1->Panels->Items[1]->Width -2;
p->Height = StatusBar1->Height-2;
p->Smooth = true;
p->Step = 1;
p->Min = 0;
p->Max = 11;
//p->Max = AnzahlZeilenExpImp*10;
StatusBar1->Repaint();
```

```
Variant Excel;

try
 {
  //Excel = GetActiveOleObject("Excel.Application");
  Excel = CreateOleObject("Excel.Application");
 }
catch(...)
 {
  Excel = CreateOleObject("Excel.Application");
 }

try
 {
  Variant WorkBooks = Excel.OlePropertyGet("WorkBooks");
  WorkBooks.OleFunction("Open",Dir + "\\files\\Vorlage_Angebot.xls");
 }
catch(...)
 {
 }
p->StepIt();
Variant ActiveWorkBook = Excel.OlePropertyGet("ActiveWorkbook");

Variant WorkSheets = Excel.OlePropertyGet("Worksheets");

//Daten im Registerblatt 1 eintragen, wie Anlagenamen, ...
Variant WorkSheet = WorkSheets.OlePropertyGet("Item", 1);
WorkSheet.OleFunction("Activate");
Variant Range = WorkSheet.OlePropertyGet("Cells", 7, 4);
Range.OlePropertySet("Value",Edit_Firma->Text);
Range = WorkSheet.OlePropertyGet("Cells", 8, 4);
Range.OlePropertySet("Value",Edit_zHd->Text);
Range = WorkSheet.OlePropertyGet("Cells", 9, 4);
Range.OlePropertySet("Value",Edit_Strasse->Text);
Range = WorkSheet.OlePropertyGet("Cells", 10, 4);
Range.OlePropertySet("Value",Edit_Ort->Text);
Range = WorkSheet.OlePropertyGet("Cells", 15, 6);
Range.OlePropertySet("Value",Edit_Nr->Text+"-"+Edit_Ver->Text+"-"+Edit_AN->Text);
Range = WorkSheet.OlePropertyGet("Cells", 16, 6);
Range.OlePropertySet("Value",Edit_Anlagenname->Text);
Range = WorkSheet.OlePropertyGet("Cells", 16, 9);
Range.OlePropertySet("Value",Edit_UZei->Text);
Range = WorkSheet.OlePropertyGet("Cells", 17, 9);
Range.OlePropertySet("Value",Edit_Datum->Text);
Range = WorkSheet.OlePropertyGet("Cells",24, 6);
Range.OlePropertySet("Value","TITEL:"+Edit_PrjName->Text);
Range = WorkSheet.OlePropertyGet("Cells",114, 4);
Range.OlePropertySet("Value",Edit_Bear->Text);

int i;
for (i=2;i<=3;i++)
 {
  WorkSheet = WorkSheets.OlePropertyGet("Item", i);
  WorkSheet.OleFunction("Activate");

  int iRow, iCol;
  try
   {
    //Beschreiben der Excelliste
    for (iRow=1; iRow <= 3; iRow++)
```

```
    {
     //Fortschrittsanzeige +1 setzen
     p->StepIt();
     for (iCol=1; iCol <=11; iCol++)
       {
        Range = WorkSheet.OlePropertyGet("Cells", iRow+1, iCol);
        switch(i)
        {
        case 1 : Range.OlePropertySet("Value", "22"); break;
        case 2 : break;
        case 3 : break;
        case 4 : break;
        case 5 : break;
        case 6 : break;
        case 7 : break;
        case 8 : break;
        case 9 : break;
        case 10 : break;
        };
       }
    }
   }
  catch(...)
   {
   ShowMessage("Fehler bei Datenexport! in Registerblatt: "+IntToStr(i)+" in Zeile: "+IntToStr(iRow)+" Spalte:
"+ IntToStr(iCol));
   }
 }
try
 {
 ActiveWorkBook.OleFunction("SaveAs", Dir + "\\Angebot.xls");
 }
catch(...)
 {
 }
Excel.OleFunction("Quit");
Excel = Unassigned;

Table1->Active = true;
//Fortschrittsanzeige auf 0 setzen
p->Position = 0;
delete p;
}
```

Header-Datei Klasse TKalkulation:

```
//---------------------------------------------------------------------------

#ifndef TKalkulationH
#define TKalkulationH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <Grids.hpp>
#include <Db.hpp>
#include <DBTables.hpp>
//---------------------------------------------------------------------------
class TKalkulation : public TForm
{
__published:       // Von der IDE verwaltete Komponenten
        TPageControl *PageControl1;
        TTabSheet *TabSheet_FG;
        TTabSheet *TabSheet_DDC;
        TTabSheet *TabSheet_SS;
        TTabSheet *TabSheet_SSBG;
        TStringGrid *StringGrid_FuG;
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label4;
        TLabel *Label5;
        TStringGrid *StringGrid_DDC;
        TStringGrid *StringGrid_SS;
        TStringGrid *StringGrid_SSBG;
        TComboBox *ComboBox_DDC;
        TComboBox *ComboBox_Zusatzauswahl;
        TDataSource *DataSource1;
        TTable *Table1;
        TQuery *Query1;
        TDatabase *Database1;
        TLabel *Label3;
        void __fastcall ComboBox_DDCChange(TObject *Sender);

private:  // Anwender-Deklarationen
        bool Stueckabfrage(void);
public:                 // Anwender-Deklarationen
        __fastcall TKalkulation(TComponent* Owner);
        void ClearKalk(void);
        void Datenuebergabe (TStringGrid* StringGrid_Baut);
        void Kalkulation (void);

        //Globale Variablen
        float Summe_List[3];
        int Module_AE,Module_AA,Module_DE,Module_DA,Sockel,CPU,Datenpkt,Basis,Divisor;
        int Summe_AE,Summe_AA,Summe_DE,Summe_DA;
};
//---------------------------------------------------------------------------
extern PACKAGE TKalkulation *Kalkulation;
//---------------------------------------------------------------------------
#endif
```

Quelldatei Klasse TKalkulation:

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "TKalkulation.h"
#include "Angebot_U.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TKalkulation *Kalkulation;

//---------------------------------------------------------------------------
// Konstruktor Klasse TKalkulation
//---------------------------------------------------------------------------
__fastcall TKalkulation::TKalkulation(TComponent* Owner)
        : TForm(Owner)
{
StringGrid_FuG->Cells[0][0] = "Stück";
StringGrid_FuG->Cells[1][0] = "Gerät";
StringGrid_FuG->Cells[2][0] = "Gerätetyp";
StringGrid_FuG->Cells[3][0] = "Gerätetyp2";
StringGrid_FuG->Cells[4][0] = "Gerätetyp3";
StringGrid_FuG->Cells[5][0] = "Hersteller";
StringGrid_FuG->Cells[6][0] = "RabattgruppeEK";
StringGrid_FuG->Cells[7][0] = "RabattgruppeVK";
StringGrid_FuG->Cells[8][0] = "ListenpreisType1";
StringGrid_FuG->Cells[9][0] = "ListenpreisType2";
StringGrid_FuG->Cells[10][0] = "ListenpreisType3";
StringGrid_FuG->Cells[11][0] = "AE";
StringGrid_FuG->Cells[12][0] = "AA";
StringGrid_FuG->Cells[13][0] = "DE";
StringGrid_FuG->Cells[14][0] = "DA";
StringGrid_FuG->Cells[15][0] = "VKEP";
StringGrid_FuG->Cells[16][0] = "VKGP";
StringGrid_FuG->Cells[17][0] = "EKEP";
StringGrid_FuG->Cells[18][0] = "EKGP";
StringGrid_FuG->Cells[19][0] = "Ausschreibungstext";

StringGrid_DDC->Cells[0][0] = "Stück";
StringGrid_DDC->Cells[1][0] = "Gerät";
StringGrid_DDC->Cells[2][0] = "Gerätetyp";
StringGrid_DDC->Cells[3][0] = "Gerätetyp2";
StringGrid_DDC->Cells[4][0] = "Gerätetyp3";
StringGrid_DDC->Cells[5][0] = "Hersteller";
StringGrid_DDC->Cells[6][0] = "RabattgruppeEK";
StringGrid_DDC->Cells[7][0] = "RabattgruppeVK";
StringGrid_DDC->Cells[8][0] = "ListenpreisType1";
StringGrid_DDC->Cells[9][0] = "ListenpreisType2";
StringGrid_DDC->Cells[10][0] = "ListenpreisType3";
StringGrid_DDC->Cells[11][0] = "AE";
StringGrid_DDC->Cells[12][0] = "AA";
StringGrid_DDC->Cells[13][0] = "DE";
StringGrid_DDC->Cells[14][0] = "DA";
StringGrid_DDC->Cells[15][0] = "VKEP";
StringGrid_DDC->Cells[16][0] = "VKGP";
StringGrid_DDC->Cells[17][0] = "EKEP";
StringGrid_DDC->Cells[18][0] = "EKGP";
```

```
StringGrid_DDC->Cells[19][0] = "Ausschreibungstext";

StringGrid_SS->Cells[0][0] = "Stück";
StringGrid_SS->Cells[1][0] = "Gerät";
StringGrid_SS->Cells[2][0] = "Gerätetyp";
StringGrid_SS->Cells[3][0] = "Gerätetyp2";
StringGrid_SS->Cells[4][0] = "Gerätetyp3";
StringGrid_SS->Cells[5][0] = "Hersteller";
StringGrid_SS->Cells[6][0] = "RabattgruppeEK";
StringGrid_SS->Cells[7][0] = "RabattgruppeVK";
StringGrid_SS->Cells[8][0] = "ListenpreisType1";
StringGrid_SS->Cells[9][0] = "ListenpreisType2";
StringGrid_SS->Cells[10][0] = "ListenpreisType3";
StringGrid_SS->Cells[11][0] = "AE";
StringGrid_SS->Cells[12][0] = "AA";
StringGrid_SS->Cells[13][0] = "DE";
StringGrid_SS->Cells[14][0] = "DA";
StringGrid_SS->Cells[15][0] = "VKEP";
StringGrid_SS->Cells[16][0] = "VKGP";
StringGrid_SS->Cells[17][0] = "EKEP";
StringGrid_SS->Cells[18][0] = "EKGP";
StringGrid_SS->Cells[19][0] = "Ausschreibungstext";

StringGrid_SSBG->Cells[0][0] = "Stück";
StringGrid_SSBG->Cells[1][0] = "Gerät";
StringGrid_SSBG->Cells[2][0] = "Gerätetyp";
StringGrid_SSBG->Cells[3][0] = "Gerätetyp2";
StringGrid_SSBG->Cells[4][0] = "Gerätetyp3";
StringGrid_SSBG->Cells[5][0] = "Hersteller";
StringGrid_SSBG->Cells[6][0] = "RabattgruppeEK";
StringGrid_SSBG->Cells[7][0] = "RabattgruppeVK";
StringGrid_SSBG->Cells[8][0] = "ListenpreisType1";
StringGrid_SSBG->Cells[9][0] = "ListenpreisType2";
StringGrid_SSBG->Cells[10][0] = "ListenpreisType3";
StringGrid_SSBG->Cells[11][0] = "AE";
StringGrid_SSBG->Cells[12][0] = "AA";
StringGrid_SSBG->Cells[13][0] = "DE";
StringGrid_SSBG->Cells[14][0] = "DA";
StringGrid_SSBG->Cells[15][0] = "VKEP";
StringGrid_SSBG->Cells[16][0] = "VKGP";
StringGrid_SSBG->Cells[17][0] = "EKEP";
StringGrid_SSBG->Cells[18][0] = "EKGP";
StringGrid_SSBG->Cells[19][0] = "Ausschreibungstext";

ComboBox_DDC->Clear();
Table1->Open();
Query1->Close();
Query1->SQL->Clear();
AnsiString ASSQLCommand = "SELECT * FROM Angebot_DB2_Aktuell WHERE Bezeichnung LIKE
'%CPU%' ORDER BY ID";
Query1->SQL->Add(ASSQLCommand);
Query1->Open();
Query1->Active = true;
Query1->First();
//ComboBox_DDC befüllen
while (!Query1->Eof)
 {
  ComboBox_DDC->Items->Add(Query1->FieldByName("Bezeichnung")->AsString);
  Query1->Next();
 }
```

```
ComboBox_Zusatzauswahl->Clear();
Table1->Open();
Query1->Close();
Query1->SQL->Clear();
ASSQLCommand = "SELECT * FROM Angebot_DB2_Aktuell WHERE BMGruppe = 'Zus' ORDER BY ID";
Query1->SQL->Add(ASSQLCommand);
Query1->Open();
Query1->Active = true;
Query1->First();
//ComboBox_Zusatzauswahl befüllen
while (!Query1->Eof)
 {
  ComboBox_Zusatzauswahl->Items->Add(Query1->FieldByName("Bezeichnung")->AsString);
  Query1->Next();
 }
}
//---------------------------------------------------------------------------
void TKalkulation::ClearKalk(void)
{
//löschen StringGrid_FuG
for (int i = 1 ; i <=StringGrid_FuG->RowCount; i++)
  for (int j = 0 ; j <=StringGrid_FuG->ColCount; j++)
     StringGrid_FuG->Cells[j][i] = "";
//löschen StringGrid_DDC
for (int i = 1 ; i <=StringGrid_DDC->RowCount; i++)
  for (int j = 0 ; j <=StringGrid_DDC->ColCount; j++)
     StringGrid_DDC->Cells[j][i] = "";
//löschen StringGrid_SS
for (int i = 1 ; i <=StringGrid_SS->RowCount; i++)
  for (int j = 0 ; j <=StringGrid_SS->ColCount; j++)
     StringGrid_SS->Cells[j][i] = "";
//löschen StringGrid_SSBG
for (int i = 1 ; i <=StringGrid_SSBG->RowCount; i++)
  for (int j = 0 ; j <=StringGrid_SSBG->ColCount; j++)
     StringGrid_SSBG->Cells[j][i] = "";

ComboBox_DDC->Clear();
Table1->Open();
Query1->Close();
Query1->SQL->Clear();
AnsiString ASSQLCommand = "SELECT * FROM Angebot_DB2_Aktuell WHERE Bezeichnung LIKE
'%CPU%' ORDER BY ID";
Query1->SQL->Add(ASSQLCommand);
Query1->Open();
Query1->Active = true;
Query1->First();
//Grobauswahl in ComboBox eintragen
while (!Query1->Eof)
 {
  ComboBox_DDC->Items->Add(Query1->FieldByName("Bezeichnung")->AsString);
  Query1->Next();
 }
ComboBox_Zusatzauswahl->Clear();
Table1->Open();
Query1->Close();
Query1->SQL->Clear();
ASSQLCommand = "SELECT * FROM Angebot_DB2_Aktuell WHERE BMGruppe = 'Zus' ORDER BY ID";
Query1->SQL->Add(ASSQLCommand);
Query1->Open();
Query1->Active = true;
```

```
Query1->First();
//ComboBox_Zusatzauswahl befüllen
while (!Query1->Eof)
 {
  ComboBox_Zusatzauswahl->Items->Add(Query1->FieldByName("Bezeichnung")->AsString);
  Query1->Next();
 }

ComboBox_DDC->ItemIndex = -1;
ComboBox_DDC->Text = "Bitte auswählen ...";
ComboBox_Zusatzauswahl->ItemIndex = -1;
ComboBox_Zusatzauswahl->Text = "Zusatzauswahl ...";
}
//---------------------------------------------------------------------------
// Datenübergabe in Klasse TKalkulation
//---------------------------------------------------------------------------
void TKalkulation::Datenuebergabe (TStringGrid* StringGrid_Baut)
{
StringGrid_FuG->RowCount = 2;
int StringGrid_FuGRowIndex=1;
for (int i =1; i <= StringGrid_Baut->RowCount; i++)
{
 //F&G in StringGrid_FuG übernehmen
 if (StringGrid_Baut->Cells[25][i] == "FuG" && StringGrid_Baut->Cells[0][i]  == "OK")
 {
 StringGrid_FuG->RowCount++;
 StringGrid_FuG->Cells[0][StringGrid_FuGRowIndex] = StringGrid_Baut->Cells[1][i];
 StringGrid_FuG->Cells[1][StringGrid_FuGRowIndex] = StringGrid_Baut->Cells[2][i];
 StringGrid_FuG->Cells[2][StringGrid_FuGRowIndex] = StringGrid_Baut->Cells[12][i];
 StringGrid_FuG->Cells[3][StringGrid_FuGRowIndex] = StringGrid_Baut->Cells[13][i];
 StringGrid_FuG->Cells[4][StringGrid_FuGRowIndex] = StringGrid_Baut->Cells[14][i];
 StringGrid_FuG->Cells[5][StringGrid_FuGRowIndex] = StringGrid_Baut->Cells[19][i];
 StringGrid_FuG->Cells[6][StringGrid_FuGRowIndex] = StringGrid_Baut->Cells[20][i];
 StringGrid_FuG->Cells[7][StringGrid_FuGRowIndex] = StringGrid_Baut->Cells[21][i];
 StringGrid_FuG->Cells[8][StringGrid_FuGRowIndex] = StringGrid_Baut->Cells[22][i];
 StringGrid_FuG->Cells[9][StringGrid_FuGRowIndex] = StringGrid_Baut->Cells[23][i];
 StringGrid_FuG->Cells[10][StringGrid_FuGRowIndex] = StringGrid_Baut->Cells[24][i];
 StringGrid_FuG->Cells[11][StringGrid_FuGRowIndex] = StringGrid_Baut->Cells[15][i];
 StringGrid_FuG->Cells[12][StringGrid_FuGRowIndex] = StringGrid_Baut->Cells[16][i];
 StringGrid_FuG->Cells[13][StringGrid_FuGRowIndex] = StringGrid_Baut->Cells[17][i];
 StringGrid_FuG->Cells[14][StringGrid_FuGRowIndex] = StringGrid_Baut->Cells[18][i];
 StringGrid_FuG->Cells[19][StringGrid_FuGRowIndex] = StringGrid_Baut->Cells[26][i];
 StringGrid_FuGRowIndex++;
 }
}
if (StringGrid_FuG->RowCount >2) StringGrid_FuG->RowCount--;

StringGrid_DDC->RowCount = 2;
int StringGrid_DDCRowIndex=1;
for (int i =1; i <= StringGrid_Baut->RowCount; i++)
{
 //DDC in StringGrid_DDC übernehmen
 if (StringGrid_Baut->Cells[25][i] == "DDC" && StringGrid_Baut->Cells[0][i]  == "OK")
 {
 StringGrid_DDC->RowCount++;
 StringGrid_DDC->Cells[0][StringGrid_DDCRowIndex] = StringGrid_Baut->Cells[1][i];
 StringGrid_DDC->Cells[1][StringGrid_DDCRowIndex] = StringGrid_Baut->Cells[2][i];
 StringGrid_DDC->Cells[2][StringGrid_DDCRowIndex] = StringGrid_Baut->Cells[12][i];
 StringGrid_DDC->Cells[3][StringGrid_DDCRowIndex] = StringGrid_Baut->Cells[13][i];
 StringGrid_DDC->Cells[4][StringGrid_DDCRowIndex] = StringGrid_Baut->Cells[14][i];
```

```
 StringGrid_DDC->Cells[5][StringGrid_DDCRowIndex] = StringGrid_Baut->Cells[19][i];
 StringGrid_DDC->Cells[6][StringGrid_DDCRowIndex] = StringGrid_Baut->Cells[20][i];
 StringGrid_DDC->Cells[7][StringGrid_DDCRowIndex] = StringGrid_Baut->Cells[21][i];
 StringGrid_DDC->Cells[8][StringGrid_DDCRowIndex] = StringGrid_Baut->Cells[22][i];
 StringGrid_DDC->Cells[9][StringGrid_DDCRowIndex] = StringGrid_Baut->Cells[23][i];
 StringGrid_DDC->Cells[10][StringGrid_DDCRowIndex] = StringGrid_Baut->Cells[24][i];
 StringGrid_DDC->Cells[11][StringGrid_DDCRowIndex] = StringGrid_Baut->Cells[15][i];
 StringGrid_DDC->Cells[12][StringGrid_DDCRowIndex] = StringGrid_Baut->Cells[16][i];
 StringGrid_DDC->Cells[13][StringGrid_DDCRowIndex] = StringGrid_Baut->Cells[17][i];
 StringGrid_DDC->Cells[14][StringGrid_DDCRowIndex] = StringGrid_Baut->Cells[18][i];
 StringGrid_DDC->Cells[19][StringGrid_DDCRowIndex] = StringGrid_Baut->Cells[26][i];
 StringGrid_DDCRowIndex++;
 }
}
if (StringGrid_DDC->RowCount >2) StringGrid_DDC->RowCount--;
StringGrid_SS->RowCount = 2;
int StringGrid_SSRowIndex=1;
for (int i =1; i <= StringGrid_Baut->RowCount; i++)
{
 //SS in StringGrid_SS übernehmen
 if (StringGrid_Baut->Cells[25][i] == "SS" && StringGrid_Baut->Cells[0][i]  == "OK")
 {
 StringGrid_SS->RowCount++;
 StringGrid_SS->Cells[0][StringGrid_SSRowIndex] = StringGrid_Baut->Cells[1][i];
 StringGrid_SS->Cells[1][StringGrid_SSRowIndex] = StringGrid_Baut->Cells[2][i];
 StringGrid_SS->Cells[2][StringGrid_SSRowIndex] = StringGrid_Baut->Cells[12][i];
 StringGrid_SS->Cells[3][StringGrid_SSRowIndex] = StringGrid_Baut->Cells[13][i];
 StringGrid_SS->Cells[4][StringGrid_SSRowIndex] = StringGrid_Baut->Cells[14][i];
 StringGrid_SS->Cells[5][StringGrid_SSRowIndex] = StringGrid_Baut->Cells[19][i];
 StringGrid_SS->Cells[6][StringGrid_SSRowIndex] = StringGrid_Baut->Cells[20][i];
 StringGrid_SS->Cells[7][StringGrid_SSRowIndex] = StringGrid_Baut->Cells[21][i];
 StringGrid_SS->Cells[8][StringGrid_SSRowIndex] = StringGrid_Baut->Cells[22][i];
 StringGrid_SS->Cells[9][StringGrid_SSRowIndex] = StringGrid_Baut->Cells[23][i];
 StringGrid_SS->Cells[10][StringGrid_SSRowIndex] = StringGrid_Baut->Cells[24][i];
 StringGrid_SS->Cells[11][StringGrid_SSRowIndex] = StringGrid_Baut->Cells[15][i];
 StringGrid_SS->Cells[12][StringGrid_SSRowIndex] = StringGrid_Baut->Cells[16][i];
 StringGrid_SS->Cells[13][StringGrid_SSRowIndex] = StringGrid_Baut->Cells[17][i];
 StringGrid_SS->Cells[14][StringGrid_SSRowIndex] = StringGrid_Baut->Cells[18][i];
 StringGrid_SS->Cells[19][StringGrid_SSRowIndex] = StringGrid_Baut->Cells[26][i];
 StringGrid_SSRowIndex++;
 }
}
if (StringGrid_SS->RowCount >2) StringGrid_SS->RowCount--;

StringGrid_SSBG->RowCount = 2;
int StringGrid_SSBGRowIndex=1;
for (int i =1; i <= StringGrid_Baut->RowCount; i++)
{
 //SSBG in StringGrid_SSBG übernehmen
 if (StringGrid_Baut->Cells[25][i] == "SSBG" && StringGrid_Baut->Cells[0][i]  == "OK")
 {
 StringGrid_SSBG->RowCount++;
 StringGrid_SSBG->Cells[0][StringGrid_SSBGRowIndex] = StringGrid_Baut->Cells[1][i];
 StringGrid_SSBG->Cells[1][StringGrid_SSBGRowIndex] = StringGrid_Baut->Cells[2][i];
 StringGrid_SSBG->Cells[2][StringGrid_SSBGRowIndex] = StringGrid_Baut->Cells[12][i];
 StringGrid_SSBG->Cells[3][StringGrid_SSBGRowIndex] = StringGrid_Baut->Cells[13][i];
 StringGrid_SSBG->Cells[4][StringGrid_SSBGRowIndex] = StringGrid_Baut->Cells[14][i];
 StringGrid_SSBG->Cells[5][StringGrid_SSBGRowIndex] = StringGrid_Baut->Cells[19][i];
 StringGrid_SSBG->Cells[6][StringGrid_SSBGRowIndex] = StringGrid_Baut->Cells[20][i];
 StringGrid_SSBG->Cells[7][StringGrid_SSBGRowIndex] = StringGrid_Baut->Cells[21][i];
```

```
  StringGrid_SSBG->Cells[8][StringGrid_SSBGRowIndex] = StringGrid_Baut->Cells[22][i];
  StringGrid_SSBG->Cells[9][StringGrid_SSBGRowIndex] = StringGrid_Baut->Cells[23][i];
  StringGrid_SSBG->Cells[10][StringGrid_SSBGRowIndex] = StringGrid_Baut->Cells[24][i];
  StringGrid_SSBG->Cells[11][StringGrid_SSBGRowIndex] = StringGrid_Baut->Cells[15][i];
  StringGrid_SSBG->Cells[12][StringGrid_SSBGRowIndex] = StringGrid_Baut->Cells[16][i];
  StringGrid_SSBG->Cells[13][StringGrid_SSBGRowIndex] = StringGrid_Baut->Cells[17][i];
  StringGrid_SSBG->Cells[14][StringGrid_SSBGRowIndex] = StringGrid_Baut->Cells[18][i];
  StringGrid_SSBG->Cells[19][StringGrid_SSBGRowIndex] = StringGrid_Baut->Cells[26][i];
  StringGrid_SSBGRowIndex++;
  }
}
if (StringGrid_SSBG->RowCount >2) StringGrid_SSBG->RowCount--;
}
//---------------------------------------------------------------------------
// Procedur Kalkulation
//---------------------------------------------------------------------------
void TKalkulation::Kalkulation (void)
{
if (StringGrid_DDC->Cells[1][1] != "" || StringGrid_FuG->Cells[1][1] != "" || StringGrid_SS->Cells[1][1] != ""
|| StringGrid_SSBG->Cells[1][1] != "")
 {
  if ((ComboBox_DDC->ItemIndex == -1) || ( Stueckabfrage()== true ))
   ShowMessage("Bitte Stückzahl und CPU Auswahl kontrollieren!");
  else
  {
   for (int ii=1;ii<=StringGrid_DDC->RowCount;ii++)
    if (StringGrid_DDC->Cells[1][ii] == "CPUBerechnung")
     {
     //löschen ab "CPUBerechnung" StringGrid_DDC
     for (int i = ii ; i <=StringGrid_DDC->RowCount; i++)
       for (int j = 0 ; j <=StringGrid_DDC->ColCount; j++)
          StringGrid_DDC->Cells[j][i] = "";
     StringGrid_DDC->RowCount = ii;
     }

   //Summe bilden
   Summe_AE=0;
   Summe_AA=0;
   Summe_DE=0;
   Summe_DA=0;
   float Summe_List1=0,Summe_List2=0,Summe_List3=0;
   float Summe_List1netto=0,Summe_List2netto=0,Summe_List3netto=0;
   float Summe_List1nettoEK=0,Summe_List2nettoEK=0,Summe_List3nettoEK=0;

   try
   {
    if (StringGrid_FuG->Cells[1][1] != "")
    {
     for (int i=1;i<=StringGrid_FuG->RowCount-1;i++)
     {
     Summe_AE    =    Summe_AE    +    (StrToInt(StringGrid_FuG->Cells[11][i])*StrToInt(StringGrid_FuG-
>Cells[0][i]));
     Summe_AA    =    Summe_AA    +    (StrToInt(StringGrid_FuG->Cells[12][i])*StrToInt(StringGrid_FuG-
>Cells[0][i]));
     Summe_DE    =    Summe_DE    +    (StrToInt(StringGrid_FuG->Cells[13][i])*StrToInt(StringGrid_FuG-
>Cells[0][i]));
     Summe_DA    =    Summe_DA    +    (StrToInt(StringGrid_FuG->Cells[14][i])*StrToInt(StringGrid_FuG-
>Cells[0][i]));
```

```
    StringGrid_FuG->Cells[15][i] = FloatToStrF((StrToFloat(StringGrid_FuG->Cells[8][i])+StrToFloat(StringGrid_FuG->Cells[9][i])+StrToFloat(StringGrid_FuG->Cells[10][i]))*F_Main->Rabkalk(StringGrid_FuG->Cells[7][i]),ffFixed,8,2);
    StringGrid_FuG->Cells[16][i] = FloatToStrF((StrToFloat(StringGrid_FuG->Cells[8][i])+StrToFloat(StringGrid_FuG->Cells[9][i])+StrToFloat(StringGrid_FuG->Cells[10][i]))*StrToInt(StringGrid_FuG->Cells[0][i])*F_Main->Rabkalk(StringGrid_FuG->Cells[7][i]),ffFixed,8,2);
    StringGrid_FuG->Cells[17][i] = FloatToStrF((StrToFloat(StringGrid_FuG->Cells[8][i])+StrToFloat(StringGrid_FuG->Cells[9][i])+StrToFloat(StringGrid_FuG->Cells[10][i]))*F_Main->RabkalkEK(StringGrid_FuG->Cells[6][i]),ffFixed,8,2);
    StringGrid_FuG->Cells[18][i] = FloatToStrF((StrToFloat(StringGrid_FuG->Cells[8][i])+StrToFloat(StringGrid_FuG->Cells[9][i])+StrToFloat(StringGrid_FuG->Cells[10][i]))*StrToInt(StringGrid_FuG->Cells[0][i])*F_Main->RabkalkEK(StringGrid_FuG->Cells[6][i]),ffFixed,8,2);
    Summe_List1 = Summe_List1 + (StrToFloat(StringGrid_FuG->Cells[8][i])*StrToInt(StringGrid_FuG->Cells[0][i]));
    Summe_List2 = Summe_List2 + (StrToFloat(StringGrid_FuG->Cells[9][i])*StrToInt(StringGrid_FuG->Cells[0][i]));
    Summe_List3 = Summe_List3 + (StrToFloat(StringGrid_FuG->Cells[10][i])*StrToInt(StringGrid_FuG->Cells[0][i]));
    Summe_List1netto = Summe_List1netto + (StrToFloat(StringGrid_FuG->Cells[8][i])*StrToInt(StringGrid_FuG->Cells[0][i])*F_Main->Rabkalk(StringGrid_FuG->Cells[7][i]));
    Summe_List2netto = Summe_List2netto + (StrToFloat(StringGrid_FuG->Cells[9][i])*StrToInt(StringGrid_FuG->Cells[0][i])*F_Main->Rabkalk(StringGrid_FuG->Cells[7][i]));
    Summe_List3netto = Summe_List3netto + (StrToFloat(StringGrid_FuG->Cells[10][i])*StrToInt(StringGrid_FuG->Cells[0][i])*F_Main->Rabkalk(StringGrid_FuG->Cells[7][i]));
    Summe_List1nettoEK = Summe_List1nettoEK + (StrToFloat(StringGrid_FuG->Cells[8][i])*StrToInt(StringGrid_FuG->Cells[0][i])*F_Main->RabkalkEK(StringGrid_FuG->Cells[6][i]));
    Summe_List2nettoEK = Summe_List2nettoEK + (StrToFloat(StringGrid_FuG->Cells[9][i])*StrToInt(StringGrid_FuG->Cells[0][i])*F_Main->RabkalkEK(StringGrid_FuG->Cells[6][i]));
    Summe_List3nettoEK = Summe_List3nettoEK + (StrToFloat(StringGrid_FuG->Cells[10][i])*StrToInt(StringGrid_FuG->Cells[0][i])*F_Main->RabkalkEK(StringGrid_FuG->Cells[6][i]));
     }
    }
   if (StringGrid_DDC->Cells[1][1] != "")
   {
    for (int i=1;i<=StringGrid_DDC->RowCount-1;i++)
    {
    Summe_AE = Summe_AE + (StrToInt(StringGrid_DDC->Cells[11][i])*StrToInt(StringGrid_DDC->Cells[0][i]));
    Summe_AA = Summe_AA + (StrToInt(StringGrid_DDC->Cells[12][i])*StrToInt(StringGrid_DDC->Cells[0][i]));
    Summe_DE = Summe_DE + (StrToInt(StringGrid_DDC->Cells[13][i])*StrToInt(StringGrid_DDC->Cells[0][i]));
    Summe_DA = Summe_DA + (StrToInt(StringGrid_DDC->Cells[14][i])*StrToInt(StringGrid_DDC->Cells[0][i]));
     }
    }
   if (StringGrid_SS->Cells[1][1] != "")
   {
    for (int i=1;i<=StringGrid_SS->RowCount-1;i++)
    {
    Summe_AE = Summe_AE + (StrToInt(StringGrid_SS->Cells[11][i])*StrToInt(StringGrid_SS->Cells[0][i]));
    Summe_AA = Summe_AA + (StrToInt(StringGrid_SS->Cells[12][i])*StrToInt(StringGrid_SS->Cells[0][i]));
    Summe_DE = Summe_DE + (StrToInt(StringGrid_SS->Cells[13][i])*StrToInt(StringGrid_SS->Cells[0][i]));
    Summe_DA = Summe_DA + (StrToInt(StringGrid_SS->Cells[14][i])*StrToInt(StringGrid_SS->Cells[0][i]));
    StringGrid_SS->Cells[15][i] = FloatToStrF((StrToFloat(StringGrid_SS->Cells[8][i])+StrToFloat(StringGrid_SS->Cells[9][i])+StrToFloat(StringGrid_SS->Cells[10][i]))*F_Main->Rabkalk(StringGrid_SS->Cells[7][i]),ffFixed,8,2);
```

```
    StringGrid_SS->Cells[16][i] = FloatToStrF((StrToFloat(StringGrid_SS->Cells[8][i])+StrToFloat(StringGrid_SS->Cells[9][i])+StrToFloat(StringGrid_SS->Cells[10][i]))*StrToInt(StringGrid_SS->Cells[0][i])*F_Main->Rabkalk(StringGrid_SS->Cells[7][i]),ffFixed,8,2);
    StringGrid_SS->Cells[17][i] = FloatToStrF((StrToFloat(StringGrid_SS->Cells[8][i])+StrToFloat(StringGrid_SS->Cells[9][i])+StrToFloat(StringGrid_SS->Cells[10][i]))*F_Main->RabkalkEK(StringGrid_SS->Cells[6][i]),ffFixed,8,2);
    StringGrid_SS->Cells[18][i] = FloatToStrF((StrToFloat(StringGrid_SS->Cells[8][i])+StrToFloat(StringGrid_SS->Cells[9][i])+StrToFloat(StringGrid_SS->Cells[10][i]))*StrToInt(StringGrid_SS->Cells[0][i])*F_Main->RabkalkEK(StringGrid_SS->Cells[6][i]),ffFixed,8,2);
    Summe_List1 = Summe_List1 + (StrToFloat(StringGrid_SS->Cells[8][i])*StrToInt(StringGrid_SS->Cells[0][i]));
    Summe_List2 = Summe_List2 + (StrToFloat(StringGrid_SS->Cells[9][i])*StrToInt(StringGrid_SS->Cells[0][i]));
    Summe_List3 = Summe_List3 + (StrToFloat(StringGrid_SS->Cells[10][i])*StrToInt(StringGrid_SS->Cells[0][i]));
    Summe_List1netto = Summe_List1netto + (StrToFloat(StringGrid_SS->Cells[8][i])*StrToInt(StringGrid_SS->Cells[0][i])*F_Main->Rabkalk(StringGrid_SS->Cells[7][i]));
    Summe_List2netto = Summe_List2netto + (StrToFloat(StringGrid_SS->Cells[9][i])*StrToInt(StringGrid_SS->Cells[0][i])*F_Main->Rabkalk(StringGrid_SS->Cells[7][i]));
    Summe_List3netto = Summe_List3netto + (StrToFloat(StringGrid_SS->Cells[10][i])*StrToInt(StringGrid_SS->Cells[0][i])*F_Main->Rabkalk(StringGrid_SS->Cells[7][i]));
    Summe_List1nettoEK = Summe_List1nettoEK + (StrToFloat(StringGrid_SS->Cells[8][i])*StrToInt(StringGrid_SS->Cells[0][i])*F_Main->RabkalkEK(StringGrid_SS->Cells[6][i]));
    Summe_List2nettoEK = Summe_List2nettoEK + (StrToFloat(StringGrid_SS->Cells[9][i])*StrToInt(StringGrid_SS->Cells[0][i])*F_Main->RabkalkEK(StringGrid_SS->Cells[6][i]));
    Summe_List3nettoEK = Summe_List3nettoEK + (StrToFloat(StringGrid_SS->Cells[10][i])*StrToInt(StringGrid_SS->Cells[0][i])*F_Main->RabkalkEK(StringGrid_SS->Cells[6][i]));
    }
   }
   if (StringGrid_SSBG->Cells[1][1] != "")
   {
    for (int i=1;i<=StringGrid_SSBG->RowCount-1;i++)
    {
    Summe_AE = Summe_AE + (StrToInt(StringGrid_SSBG->Cells[11][i])*StrToInt(StringGrid_SSBG->Cells[0][i]));
    Summe_AA = Summe_AA + (StrToInt(StringGrid_SSBG->Cells[12][i])*StrToInt(StringGrid_SSBG->Cells[0][i]));
    Summe_DE = Summe_DE + (StrToInt(StringGrid_SSBG->Cells[13][i])*StrToInt(StringGrid_SSBG->Cells[0][i]));
    Summe_DA = Summe_DA + (StrToInt(StringGrid_SSBG->Cells[14][i])*StrToInt(StringGrid_SSBG->Cells[0][i]));
    StringGrid_SSBG->Cells[15][i] = FloatToStrF((StrToFloat(StringGrid_SSBG->Cells[8][i])+StrToFloat(StringGrid_SSBG->Cells[9][i])+StrToFloat(StringGrid_SSBG->Cells[10][i]))*F_Main->Rabkalk(StringGrid_SSBG->Cells[7][i]),ffFixed,8,2);
    StringGrid_SSBG->Cells[16][i] = FloatToStrF((StrToFloat(StringGrid_SSBG->Cells[8][i])+StrToFloat(StringGrid_SSBG->Cells[9][i])+StrToFloat(StringGrid_SSBG->Cells[10][i]))*StrToInt(StringGrid_SSBG->Cells[0][i])*F_Main->Rabkalk(StringGrid_SSBG->Cells[7][i]),ffFixed,8,2);
    StringGrid_SSBG->Cells[17][i] = FloatToStrF((StrToFloat(StringGrid_SSBG->Cells[8][i])+StrToFloat(StringGrid_SSBG->Cells[9][i])+StrToFloat(StringGrid_SSBG->Cells[10][i]))*F_Main->RabkalkEK(StringGrid_SSBG->Cells[6][i]),ffFixed,8,2);
    StringGrid_SSBG->Cells[18][i] = FloatToStrF((StrToFloat(StringGrid_SSBG->Cells[8][i])+StrToFloat(StringGrid_SSBG->Cells[9][i])+StrToFloat(StringGrid_SSBG->Cells[10][i]))*StrToInt(StringGrid_SSBG->Cells[0][i])*F_Main->RabkalkEK(StringGrid_SSBG->Cells[6][i]),ffFixed,8,2);
    Summe_List1 = Summe_List1 + (StrToFloat(StringGrid_SSBG->Cells[8][i])*StrToInt(StringGrid_SSBG->Cells[0][i]));
```

```
    Summe_List2 = Summe_List2 + (StrToFloat(StringGrid_SSBG->Cells[9][i])*StrToInt(StringGrid_SSBG->Cells[0][i]));
    Summe_List3 = Summe_List3 + (StrToFloat(StringGrid_SSBG->Cells[10][i])*StrToInt(StringGrid_SSBG->Cells[0][i]));
    Summe_List1netto = Summe_List1netto + (StrToFloat(StringGrid_SSBG->Cells[8][i])*StrToInt(StringGrid_SSBG->Cells[0][i])*F_Main->Rabkalk(StringGrid_SSBG->Cells[7][i]));
    Summe_List2netto = Summe_List2netto + (StrToFloat(StringGrid_SSBG->Cells[9][i])*StrToInt(StringGrid_SSBG->Cells[0][i])*F_Main->Rabkalk(StringGrid_SSBG->Cells[7][i]));
    Summe_List3netto = Summe_List3netto + (StrToFloat(StringGrid_SSBG->Cells[10][i])*StrToInt(StringGrid_SSBG->Cells[0][i])*F_Main->Rabkalk(StringGrid_SSBG->Cells[7][i]));
    Summe_List1nettoEK = Summe_List1nettoEK + (StrToFloat(StringGrid_SSBG->Cells[8][i])*StrToInt(StringGrid_SSBG->Cells[0][i])*F_Main->RabkalkEK(StringGrid_SSBG->Cells[6][i]));
    Summe_List2nettoEK = Summe_List2nettoEK + (StrToFloat(StringGrid_SSBG->Cells[9][i])*StrToInt(StringGrid_SSBG->Cells[0][i])*F_Main->RabkalkEK(StringGrid_SSBG->Cells[6][i]));
    Summe_List3nettoEK = Summe_List3nettoEK + (StrToFloat(StringGrid_SSBG->Cells[10][i])*StrToInt(StringGrid_SSBG->Cells[0][i])*F_Main->RabkalkEK(StringGrid_SSBG->Cells[6][i]));
     }
    }
   }
   catch(...)
    {
    ShowMessage("Bitte Stückzahl, Preise und AE,AA,DE,DA auf Plausibilität prüfen!");
    }

   StringGrid_DDC->RowCount++;
   StringGrid_DDC->Cells[1][StringGrid_DDC->RowCount-1] = "CPUBerechnung";

   Module_AE=0;
   Module_AA=0;
   Module_DE=0;
   Module_DA=0;
   Sockel=0;
   CPU=0;
   Datenpkt=0;
   Basis=0;
   Divisor=0;

   //Division durch x(8) - wenn Rest + 1
   div_t x;

   // Zuerst CPU Auswahl - Berechnung über Datenpunkte oder Modulanzahl
   int DivisorZus;
   AnsiString BezeichnungZus;

   //Berechnung Anzahl Datenpunkte
   Datenpkt = Summe_AE + Summe_AA + Summe_DE + Summe_DA;

   Table1->Open();
   Query1->Close();
   Query1->SQL->Clear();
   AnsiString ASSQLCommand1 = "SELECT * FROM Angebot_DB2_Aktuell WHERE Bezeichnung = '"+ComboBox_DDC->Text+"'";
   Query1->SQL->Add(ASSQLCommand1);
   Query1->Open();
   Query1->Active = true;
   Query1->First();

   StringGrid_DDC->RowCount++;
   int i = StringGrid_DDC->RowCount-1;
   StringGrid_DDC->Cells[1][i] = Query1->FieldByName("Bezeichnung")->AsString;
```

```
  StringGrid_DDC->Cells[2][i] = Query1->FieldByName("Gerätetyp")->AsString;
  StringGrid_DDC->Cells[3][i] = Query1->FieldByName("Gerätetyp2")->AsString;
  StringGrid_DDC->Cells[4][i] = Query1->FieldByName("Gerätetyp3")->AsString;
  StringGrid_DDC->Cells[5][i] = Query1->FieldByName("Hersteller")->AsString;
  StringGrid_DDC->Cells[6][i] = Query1->FieldByName("RabattgruppeEK")->AsString;
  StringGrid_DDC->Cells[7][i] = Query1->FieldByName("RabattgruppeVK")->AsString;
  StringGrid_DDC->Cells[8][i] = Query1->FieldByName("ListenpreisType1")->AsString;
  StringGrid_DDC->Cells[9][i] = Query1->FieldByName("ListenpreisType2")->AsString;
  StringGrid_DDC->Cells[10][i] = Query1->FieldByName("ListenpreisType3")->AsString;
  StringGrid_DDC->Cells[19][i] = Query1->FieldByName("Ausschreibungstext")->AsString;

  if (ComboBox_Zusatzauswahl->ItemIndex >=0)
     StringGrid_DDC->Cells[3][StringGrid_DDC->RowCount-1] = BezeichnungZus;

  int CPUZeile = StringGrid_DDC->RowCount-1;
  Basis = Datenpkt;
  Divisor = Query1->FieldByName("DatenpktXLWeb")->AsInteger;
  bool BerechnungUeberDatenpkte = Query1->FieldByName("Datenpunkte")->AsBoolean;
  int AEModultypID = Query1->FieldByName("AE")->AsInteger;
  int AAModultypID = Query1->FieldByName("AA")->AsInteger;
  int DEModultypID = Query1->FieldByName("DE")->AsInteger;
  int DAModultypID = Query1->FieldByName("DA")->AsInteger;

  //XL50CPU ja/nein?
  bool XL50CPU;
  int Ver = CompareStr(Query1->FieldByName("Bezeichnung")->AsString,"XL 50-D");
  int Ver1 = CompareStr(Query1->FieldByName("Bezeichnung")->AsString,"XL 50-W");
  if (Ver == 0 || Ver1 == 0) XL50CPU = true;
  else XL50CPU = false;

  //RomutecCPU ja/nein?   Geht noch nicht
  bool RomutecCPU;
  AnsiString string1=Query1->FieldByName("Bezeichnung")->AsString;
  int g = string1.Pos("Romutec");
  if (g > 0)
  {
  RomutecCPU = true;
  }
  else RomutecCPU = false;

  // Ausahl der Module

  Table1->Open();
  Query1->Close();
  Query1->SQL->Clear();
  ASSQLCommand1 = "SELECT * FROM Angebot_DB2_Aktuell WHERE ID =
'"+IntToStr(AEModultypID)+"'";
  Query1->SQL->Add(ASSQLCommand1);
  Query1->Open();
  Query1->Active = true;
  Query1->First();
  int h=0;
  if (XL50CPU == true) h = 8;
  x = div(Summe_AE-h,Query1->FieldByName("AE")->AsInteger);
  Module_AE = x.quot;
  if (x.rem > 0) Module_AE++;
  if (Module_AE > 0)
  {
  StringGrid_DDC->RowCount++;
  int i = StringGrid_DDC->RowCount-1;
```

```
  StringGrid_DDC->Cells[0][i] = Module_AE;
  StringGrid_DDC->Cells[1][i] = Query1->FieldByName("Bezeichnung")->AsString;
  StringGrid_DDC->Cells[2][i] = Query1->FieldByName("Gerätetyp")->AsString;
  StringGrid_DDC->Cells[3][i] = Query1->FieldByName("Gerätetyp2")->AsString;
  StringGrid_DDC->Cells[4][i] = Query1->FieldByName("Gerätetyp3")->AsString;
  StringGrid_DDC->Cells[5][i] = Query1->FieldByName("Hersteller")->AsString;
  StringGrid_DDC->Cells[6][i] = Query1->FieldByName("RabattgruppeEK")->AsString;
  StringGrid_DDC->Cells[7][i] = Query1->FieldByName("RabattgruppeVK")->AsString;
  StringGrid_DDC->Cells[8][i] = Query1->FieldByName("ListenpreisType1")->AsString;
  StringGrid_DDC->Cells[9][i] = Query1->FieldByName("ListenpreisType2")->AsString;
  StringGrid_DDC->Cells[10][i] = Query1->FieldByName("ListenpreisType3")->AsString;
  StringGrid_DDC->Cells[15][i]                =                FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC->Cells[10][i]))*F_Main-
>Rabkalk(StringGrid_DDC->Cells[7][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[16][i]                =                FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC-
>Cells[10][i]))*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->Rabkalk(StringGrid_DDC-
>Cells[7][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[17][i]                =                FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC->Cells[10][i]))*F_Main-
>RabkalkEK(StringGrid_DDC->Cells[6][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[18][i]                =                FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC-
>Cells[10][i]))*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->RabkalkEK(StringGrid_DDC-
>Cells[6][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[19][i] = Query1->FieldByName("Ausschreibungstext")->AsString;

  }

  Table1->Open();
  Query1->Close();
  Query1->SQL->Clear();
  ASSQLCommand1      =      "SELECT      *      FROM      Angebot_DB2_Aktuell      WHERE      ID      =
'"+IntToStr(AAModultypID)+"'";
  Query1->SQL->Add(ASSQLCommand1);
  Query1->Open();
  Query1->Active = true;
  Query1->First();
  h=0;
  if (XL50CPU == true) h = 4;
  x = div(Summe_AA-h,Query1->FieldByName("AA")->AsInteger);
  Module_AA = x.quot;
  if (x.rem > 0) Module_AA++;
  if (Module_AA > 0)
  {
  StringGrid_DDC->RowCount++;
  int i = StringGrid_DDC->RowCount-1;
  StringGrid_DDC->Cells[0][i] = Module_AA;
  StringGrid_DDC->Cells[1][i] = Query1->FieldByName("Bezeichnung")->AsString;
  StringGrid_DDC->Cells[2][i] = Query1->FieldByName("Gerätetyp")->AsString;
  StringGrid_DDC->Cells[3][i] = Query1->FieldByName("Gerätetyp2")->AsString;
  StringGrid_DDC->Cells[4][i] = Query1->FieldByName("Gerätetyp3")->AsString;
  StringGrid_DDC->Cells[5][i] = Query1->FieldByName("Hersteller")->AsString;
  StringGrid_DDC->Cells[6][i] = Query1->FieldByName("RabattgruppeEK")->AsString;
  StringGrid_DDC->Cells[7][i] = Query1->FieldByName("RabattgruppeVK")->AsString;
  StringGrid_DDC->Cells[8][i] = Query1->FieldByName("ListenpreisType1")->AsString;
  StringGrid_DDC->Cells[9][i] = Query1->FieldByName("ListenpreisType2")->AsString;
  StringGrid_DDC->Cells[10][i] = Query1->FieldByName("ListenpreisType3")->AsString;
```

```
  StringGrid_DDC->Cells[15][i]  =  FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC->Cells[10][i]))*F_Main-
>Rabkalk(StringGrid_DDC->Cells[7][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[16][i]  =  FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC-
>Cells[10][i]))*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->Rabkalk(StringGrid_DDC-
>Cells[7][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[17][i]  =  FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC->Cells[10][i]))*F_Main-
>RabkalkEK(StringGrid_DDC->Cells[6][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[18][i]  =  FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC-
>Cells[10][i]))*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->RabkalkEK(StringGrid_DDC-
>Cells[6][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[19][i] = Query1->FieldByName("Ausschreibungstext")->AsString;
  }

  Table1->Open();
  Query1->Close();
  Query1->SQL->Clear();
  ASSQLCommand1  =  "SELECT  *  FROM  Angebot_DB2_Aktuell  WHERE  ID  =
'"+IntToStr(DAModultypID)+"'";
  Query1->SQL->Add(ASSQLCommand1);
  Query1->Open();
  Query1->Active = true;
  Query1->First();
  h=0;
  if (XL50CPU == true) h = 6;
  x = div(Summe_DA-h,Query1->FieldByName("DA")->AsInteger);
  int DERomutec = Query1->FieldByName("DE")->AsInteger;
  Module_DA = x.quot;
  if (x.rem > 0) Module_DA++;
  if (Module_DA > 0)
  {
  StringGrid_DDC->RowCount++;
  int i = StringGrid_DDC->RowCount-1;
  StringGrid_DDC->Cells[0][i] = Module_DA;
  StringGrid_DDC->Cells[1][i] = Query1->FieldByName("Bezeichnung")->AsString;
  StringGrid_DDC->Cells[2][i] = Query1->FieldByName("Gerätetyp")->AsString;
  StringGrid_DDC->Cells[3][i] = Query1->FieldByName("Gerätetyp2")->AsString;
  StringGrid_DDC->Cells[4][i] = Query1->FieldByName("Gerätetyp3")->AsString;
  StringGrid_DDC->Cells[5][i] = Query1->FieldByName("Hersteller")->AsString;
  StringGrid_DDC->Cells[6][i] = Query1->FieldByName("RabattgruppeEK")->AsString;
  StringGrid_DDC->Cells[7][i] = Query1->FieldByName("RabattgruppeVK")->AsString;
  StringGrid_DDC->Cells[8][i] = Query1->FieldByName("ListenpreisType1")->AsString;
  StringGrid_DDC->Cells[9][i] = Query1->FieldByName("ListenpreisType2")->AsString;
  StringGrid_DDC->Cells[10][i] = Query1->FieldByName("ListenpreisType3")->AsString;
  StringGrid_DDC->Cells[15][i]  =  FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC->Cells[10][i]))*F_Main-
>Rabkalk(StringGrid_DDC->Cells[7][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[16][i]  =  FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC-
>Cells[10][i]))*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->Rabkalk(StringGrid_DDC-
>Cells[7][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[17][i]  =  FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC->Cells[10][i]))*F_Main-
>RabkalkEK(StringGrid_DDC->Cells[6][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[18][i]  =  FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC-
```

```
>Cells[10][i]))*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->RabkalkEK(StringGrid_DDC-
>Cells[6][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[19][i] = Query1->FieldByName("Ausschreibungstext")->AsString;
  }

  Table1->Open();
  Query1->Close();
  Query1->SQL->Clear();
  ASSQLCommand1 = "SELECT * FROM Angebot_DB2_Aktuell WHERE ID =
'"+IntToStr(DEModultypID)+"'";
  Query1->SQL->Add(ASSQLCommand1);
  Query1->Open();
  Query1->Active = true;
  Query1->First();
  h=0;
  if (XL50CPU == true) h = 4;
  if (RomutecCPU == true) h = DERomutec*Module_DA;

  x = div(Summe_DE-h,Query1->FieldByName("DE")->AsInteger);
  Module_DE = x.quot;
  if (x.rem > 0) Module_DE++;
  if (Module_DE > 0)
  {
  StringGrid_DDC->RowCount++;
  int i = StringGrid_DDC->RowCount-1;
  StringGrid_DDC->Cells[0][i] = Module_DE;
  StringGrid_DDC->Cells[1][i] = Query1->FieldByName("Bezeichnung")->AsString;
  StringGrid_DDC->Cells[2][i] = Query1->FieldByName("Gerätetyp")->AsString;
  StringGrid_DDC->Cells[3][i] = Query1->FieldByName("Gerätetyp2")->AsString;
  StringGrid_DDC->Cells[4][i] = Query1->FieldByName("Gerätetyp3")->AsString;
  StringGrid_DDC->Cells[5][i] = Query1->FieldByName("Hersteller")->AsString;
  StringGrid_DDC->Cells[6][i] = Query1->FieldByName("RabattgruppeEK")->AsString;
  StringGrid_DDC->Cells[7][i] = Query1->FieldByName("RabattgruppeVK")->AsString;
  StringGrid_DDC->Cells[8][i] = Query1->FieldByName("ListenpreisType1")->AsString;
  StringGrid_DDC->Cells[9][i] = Query1->FieldByName("ListenpreisType2")->AsString;
  StringGrid_DDC->Cells[10][i] = Query1->FieldByName("ListenpreisType3")->AsString;
  StringGrid_DDC->Cells[15][i] = FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC->Cells[10][i]))*F_Main-
>Rabkalk(StringGrid_DDC->Cells[7][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[16][i] = FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC-
>Cells[10][i]))*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->Rabkalk(StringGrid_DDC-
>Cells[7][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[17][i] = FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC->Cells[10][i]))*F_Main-
>RabkalkEK(StringGrid_DDC->Cells[6][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[18][i] = FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC-
>Cells[10][i]))*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->RabkalkEK(StringGrid_DDC-
>Cells[6][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[19][i] = Query1->FieldByName("Ausschreibungstext")->AsString;
  }


  //Anzahl Sockel
  Sockel = Module_AE + Module_AA + Module_DE + Module_DA;
  int SockelAEAA = Module_AE + Module_AA;

  if (SockelAEAA > 0)
  {
```

```
  Table1->Open();
  Query1->Close();
  Query1->SQL->Clear();
  ASSQLCommand1 = "SELECT * FROM Angebot_DB2_Aktuell WHERE ID = 32";
  Query1->SQL->Add(ASSQLCommand1);
  Query1->Open();
  Query1->Active = true;
  Query1->First();
  StringGrid_DDC->RowCount++;
  int i = StringGrid_DDC->RowCount-1;
  StringGrid_DDC->Cells[0][i] = SockelAEAA;
  StringGrid_DDC->Cells[1][i] = Query1->FieldByName("Bezeichnung")->AsString;
  StringGrid_DDC->Cells[2][i] = Query1->FieldByName("Gerätetyp")->AsString;
  StringGrid_DDC->Cells[3][i] = Query1->FieldByName("Gerätetyp2")->AsString;
  StringGrid_DDC->Cells[4][i] = Query1->FieldByName("Gerätetyp3")->AsString;
  StringGrid_DDC->Cells[5][i] = Query1->FieldByName("Hersteller")->AsString;
  StringGrid_DDC->Cells[6][i] = Query1->FieldByName("RabattgruppeEK")->AsString;
  StringGrid_DDC->Cells[7][i] = Query1->FieldByName("RabattgruppeVK")->AsString;
  StringGrid_DDC->Cells[8][i] = Query1->FieldByName("ListenpreisType1")->AsString;
  StringGrid_DDC->Cells[9][i] = Query1->FieldByName("ListenpreisType2")->AsString;
  StringGrid_DDC->Cells[10][i] = Query1->FieldByName("ListenpreisType3")->AsString;
  StringGrid_DDC->Cells[15][i]                =                FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC->Cells[10][i]))*F_Main-
>Rabkalk(StringGrid_DDC->Cells[7][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[16][i]                =                FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC-
>Cells[10][i]))*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->Rabkalk(StringGrid_DDC-
>Cells[7][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[17][i]                =                FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC->Cells[10][i]))*F_Main-
>RabkalkEK(StringGrid_DDC->Cells[6][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[18][i]                =                FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC-
>Cells[10][i]))*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->RabkalkEK(StringGrid_DDC-
>Cells[6][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[19][i] = Query1->FieldByName("Ausschreibungstext")->AsString;
  }

  if (Module_DE > 0)
  {
  Table1->Open();
  Query1->Close();
  Query1->SQL->Clear();
  ASSQLCommand1 = "SELECT * FROM Angebot_DB2_Aktuell WHERE ID = 33";
  Query1->SQL->Add(ASSQLCommand1);
  Query1->Open();
  Query1->Active = true;
  Query1->First();
  StringGrid_DDC->RowCount++;
  int i = StringGrid_DDC->RowCount-1;
  StringGrid_DDC->Cells[0][i] = Module_DE;
  StringGrid_DDC->Cells[1][i] = Query1->FieldByName("Bezeichnung")->AsString;
  StringGrid_DDC->Cells[2][i] = Query1->FieldByName("Gerätetyp")->AsString;
  StringGrid_DDC->Cells[3][i] = Query1->FieldByName("Gerätetyp2")->AsString;
  StringGrid_DDC->Cells[4][i] = Query1->FieldByName("Gerätetyp3")->AsString;
  StringGrid_DDC->Cells[5][i] = Query1->FieldByName("Hersteller")->AsString;
  StringGrid_DDC->Cells[6][i] = Query1->FieldByName("RabattgruppeEK")->AsString;
  StringGrid_DDC->Cells[7][i] = Query1->FieldByName("RabattgruppeVK")->AsString;
  StringGrid_DDC->Cells[8][i] = Query1->FieldByName("ListenpreisType1")->AsString;
  StringGrid_DDC->Cells[9][i] = Query1->FieldByName("ListenpreisType2")->AsString;
```

```
  StringGrid_DDC->Cells[10][i] = Query1->FieldByName("ListenpreisType3")->AsString;
  StringGrid_DDC->Cells[15][i]                =                FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC->Cells[10][i]))*F_Main-
>Rabkalk(StringGrid_DDC->Cells[7][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[16][i]                =                FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC-
>Cells[10][i]))*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->Rabkalk(StringGrid_DDC-
>Cells[7][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[17][i]                =                FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC->Cells[10][i]))*F_Main-
>RabkalkEK(StringGrid_DDC->Cells[6][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[18][i]                =                FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC-
>Cells[10][i]))*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->RabkalkEK(StringGrid_DDC-
>Cells[6][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[19][i] = Query1->FieldByName("Ausschreibungstext")->AsString;
  }

  if (Module_DA > 0)
  {
  Table1->Open();
  Query1->Close();
  Query1->SQL->Clear();
  ASSQLCommand1 = "SELECT * FROM Angebot_DB2_Aktuell WHERE ID = 34";
  Query1->SQL->Add(ASSQLCommand1);
  Query1->Open();
  Query1->Active = true;
  Query1->First();
  StringGrid_DDC->RowCount++;
  int i = StringGrid_DDC->RowCount-1;
  StringGrid_DDC->Cells[0][i] = Module_DA;
  StringGrid_DDC->Cells[1][i] = Query1->FieldByName("Bezeichnung")->AsString;
  StringGrid_DDC->Cells[2][i] = Query1->FieldByName("Gerätetyp")->AsString;
  StringGrid_DDC->Cells[3][i] = Query1->FieldByName("Gerätetyp2")->AsString;
  StringGrid_DDC->Cells[4][i] = Query1->FieldByName("Gerätetyp3")->AsString;
  StringGrid_DDC->Cells[5][i] = Query1->FieldByName("Hersteller")->AsString;
  StringGrid_DDC->Cells[6][i] = Query1->FieldByName("RabattgruppeEK")->AsString;
  StringGrid_DDC->Cells[7][i] = Query1->FieldByName("RabattgruppeVK")->AsString;
  StringGrid_DDC->Cells[8][i] = Query1->FieldByName("ListenpreisType1")->AsString;
  StringGrid_DDC->Cells[9][i] = Query1->FieldByName("ListenpreisType2")->AsString;
  StringGrid_DDC->Cells[10][i] = Query1->FieldByName("ListenpreisType3")->AsString;
  StringGrid_DDC->Cells[15][i]                =                FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC->Cells[10][i]))*F_Main-
>Rabkalk(StringGrid_DDC->Cells[7][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[16][i]                =                FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC-
>Cells[10][i]))*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->Rabkalk(StringGrid_DDC-
>Cells[7][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[17][i]                =                FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC->Cells[10][i]))*F_Main-
>RabkalkEK(StringGrid_DDC->Cells[6][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[18][i]                =                FloatToStrF((StrToFloat(StringGrid_DDC-
>Cells[8][i])+StrToFloat(StringGrid_DDC->Cells[9][i])+StrToFloat(StringGrid_DDC-
>Cells[10][i]))*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->RabkalkEK(StringGrid_DDC-
>Cells[6][i]),ffFixed,8,2);
  StringGrid_DDC->Cells[19][i] = Query1->FieldByName("Ausschreibungstext")->AsString;
  }

  //Berechnung Anzahl notwendiger CPUs erst hier, da manche über Sockel berechnet
  if (BerechnungUeberDatenpkte == False)
```

```
  {
   Basis = Sockel;
  // Divisor = 16;
  }

 if (Basis > 0)
  {
   x = div(Basis,Divisor);
   CPU = x.quot;
   if (x.rem > 0) CPU++;
   StringGrid_DDC->Cells[0][CPUZeile] = CPU;
   StringGrid_DDC->Cells[15][CPUZeile] = FloatToStrF((StrToFloat(StringGrid_DDC->Cells[8][CPUZeile])+StrToFloat(StringGrid_DDC->Cells[9][CPUZeile])+StrToFloat(StringGrid_DDC->Cells[10][CPUZeile]))*F_Main->Rabkalk(StringGrid_DDC->Cells[7][CPUZeile]),ffFixed,8,2);
   StringGrid_DDC->Cells[16][CPUZeile] = FloatToStrF((StrToFloat(StringGrid_DDC->Cells[8][CPUZeile])+StrToFloat(StringGrid_DDC->Cells[9][CPUZeile])+StrToFloat(StringGrid_DDC->Cells[10][CPUZeile]))*StrToInt(StringGrid_DDC->Cells[0][CPUZeile])*F_Main->Rabkalk(StringGrid_DDC->Cells[7][CPUZeile]),ffFixed,8,2);
   StringGrid_DDC->Cells[17][CPUZeile] = FloatToStrF((StrToFloat(StringGrid_DDC->Cells[8][CPUZeile])+StrToFloat(StringGrid_DDC->Cells[9][CPUZeile])+StrToFloat(StringGrid_DDC->Cells[10][CPUZeile]))*F_Main->RabkalkEK(StringGrid_DDC->Cells[6][CPUZeile]),ffFixed,8,2);
   StringGrid_DDC->Cells[18][CPUZeile] = FloatToStrF((StrToFloat(StringGrid_DDC->Cells[8][CPUZeile])+StrToFloat(StringGrid_DDC->Cells[9][CPUZeile])+StrToFloat(StringGrid_DDC->Cells[10][CPUZeile]))*StrToInt(StringGrid_DDC->Cells[0][CPUZeile])*F_Main->RabkalkEK(StringGrid_DDC->Cells[6][CPUZeile]),ffFixed,8,2);
  }

 try
  {
  int i;
  if (StringGrid_DDC->Cells[1][1] != "")
  i = 1;
  else i = 2;
  for (i;i<=StringGrid_DDC->RowCount-1;i++)
   {
   if (StringGrid_DDC->Cells[1][i] != "CPUBerechnung")
    {
     Summe_List1 = Summe_List1 + (StrToFloat(StringGrid_DDC->Cells[8][i])*StrToInt(StringGrid_DDC->Cells[0][i]));
     Summe_List2 = Summe_List2 + (StrToFloat(StringGrid_DDC->Cells[9][i])*StrToInt(StringGrid_DDC->Cells[0][i]));
     Summe_List3 = Summe_List3 + (StrToFloat(StringGrid_DDC->Cells[10][i])*StrToInt(StringGrid_DDC->Cells[0][i]));
     Summe_List1netto = Summe_List1netto + (StrToFloat(StringGrid_DDC->Cells[8][i])*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->Rabkalk(StringGrid_DDC->Cells[7][i]));
     Summe_List2netto = Summe_List2netto + (StrToFloat(StringGrid_DDC->Cells[9][i])*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->Rabkalk(StringGrid_DDC->Cells[7][i]));
     Summe_List3netto = Summe_List3netto + (StrToFloat(StringGrid_DDC->Cells[10][i])*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->Rabkalk(StringGrid_DDC->Cells[7][i]));
     Summe_List1nettoEK = Summe_List1nettoEK + (StrToFloat(StringGrid_DDC->Cells[8][i])*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->RabkalkEK(StringGrid_DDC->Cells[6][i]));
     Summe_List2nettoEK = Summe_List2nettoEK + (StrToFloat(StringGrid_DDC->Cells[9][i])*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->RabkalkEK(StringGrid_DDC->Cells[6][i]));
     Summe_List3nettoEK = Summe_List3nettoEK + (StrToFloat(StringGrid_DDC->Cells[10][i])*StrToInt(StringGrid_DDC->Cells[0][i])*F_Main->RabkalkEK(StringGrid_DDC->Cells[6][i]));
    }
   }
  }
 catch(...)
  {
```

```
   ShowMessage("Bitte Preise auf Plausibilität prüfen!");
   }

  //Anzeige der Summe
  StringGrid_DDC->RowCount++;
  StringGrid_DDC->Cells[1][StringGrid_DDC->RowCount-1] = "Gesamt";

  StringGrid_DDC->Cells[11][StringGrid_DDC->RowCount-1] = IntToStr(Summe_AE);
  StringGrid_DDC->Cells[12][StringGrid_DDC->RowCount-1] = IntToStr(Summe_AA);
  StringGrid_DDC->Cells[13][StringGrid_DDC->RowCount-1] = IntToStr(Summe_DE);
  StringGrid_DDC->Cells[14][StringGrid_DDC->RowCount-1] = IntToStr(Summe_DA);
  StringGrid_DDC->Cells[8][StringGrid_DDC->RowCount-1] = FloatToStrF(Summe_List1,ffFixed,8,2);
  StringGrid_DDC->Cells[9][StringGrid_DDC->RowCount-1] = FloatToStrF(Summe_List2,ffFixed,8,2);
  StringGrid_DDC->Cells[10][StringGrid_DDC->RowCount-1] = FloatToStrF(Summe_List3,ffFixed,8,2);
  Summe_List[0] = Summe_List1+Summe_List2+Summe_List3;
  Summe_List[1] = Summe_List1nettoEK+Summe_List2nettoEK+Summe_List3nettoEK;
  Summe_List[2] = Summe_List1netto+Summe_List2netto+Summe_List3netto;
 }
 }
}
//---------------------------------------------------------------------------
// Procedur Stückabfrage (Return true = Fehler, false = OK)
//---------------------------------------------------------------------------
bool TKalkulation::Stueckabfrage (void)
{
bool Fehler = false;
for (int i = 1 ; i <=StringGrid_FuG->RowCount; i++)
 {
 if (StringGrid_FuG->Cells[0][i] == "" && StringGrid_FuG->Cells[1][i] != "")
   Fehler = true;
 }
for (int i = 1 ; i <=StringGrid_DDC->RowCount; i++)
 {
 if (StringGrid_DDC->Cells[0][i] == "" && StringGrid_DDC->Cells[1][i] != ""
    && StringGrid_DDC->Cells[1][i] != "Gesamt" && StringGrid_DDC->Cells[1][i] != "CPUBerechnung")
   Fehler = true;
 }
for (int i = 1 ; i <=StringGrid_SS->RowCount; i++)
 {
 if (StringGrid_SS->Cells[0][i] == "" && StringGrid_SS->Cells[1][i] != "")
   Fehler = true;
 }
for (int i = 1 ; i <=StringGrid_SSBG->RowCount; i++)
 {
 if (StringGrid_SSBG->Cells[0][i] == "" && StringGrid_SSBG->Cells[1][i] != "")
   Fehler = true;
 }

return (Fehler);
}
//---------------------------------------------------------------------------
void __fastcall TKalkulation::ComboBox_DDCChange(TObject *Sender)
{
if (ComboBox_DDC->ItemIndex >=0)
  ComboBox_Zusatzauswahl->Enabled = true;
  else
  ComboBox_Zusatzauswahl->Enabled = false;
}
```

Erklärung

Ich erkläre, dass ich die vorliegende Diplomarbeit selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Brunn am Gebirge, 01.06.2009 _____